iSBC 941™
INDUSTRIAL DIGITAL PROCESSOR USER'S GUIDE

Manual Order Number: 9803077-02

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

The following are trademarks of Intel Corporation and may be used only to describe Intel products:

i	iSBC	Multimodule
ICE	Library Manager	PROMPT
iCS	MCS	Promware
Insite	Megachassis	RMX
Intel	Micromap	UPI
Intellec	Multibus	μScope

and the combination of ICE, iCS, iSBC, MCS, or RMX and a numerical suffix.

A99/0879/10K DD

PREFACE

This manual provides general information, specifications, interfacing information, and operating instructions for the iSBC 941 Industrial Digital Processor. Additional information is available in the following documents:

- *Intel iSBC 569 Intelligent Digital Controller Board Hardware Reference Manual*, Order Number 9800845.
- *Closed Loop Control Using the Intel iSBC 569/941 Processors*, Application Note AP-60.

CONTENTS

TABLES

ILLUSTRATIONS

CHAPTER 1
GENERAL INFORMATION

1-1. INTRODUCTION

The iSBC 941 Industrial Digital Processor (IDP) is a single chip microcomputer, especially designed for industrial control applications. The iSBC 941 IDP device, packaged in a 40-pin DIP, can execute nine dedicated I/O algorithms and eight common data manipulation commands. The iSBC 941 IDP functions as an intelligent slave processor to a host CPU, in MCS-80, MCS-85, and MCS-86 based systems. It can be readily installed onto the iSBC 80/30 Single Board Computer and the iSBC 569 Intelligent Digital Controller.

This manual describes the iSBC 941 IDP and provides instruction for its use. Chapter 1 provides general information and specifications. Chapter 2 provides interfacing information and a brief introduction to commands. Chapter 3 provides a more detailed description of system operation. Appendices A, B and C include tables which are used in conjunction with the text.

1-2. DESCRIPTION

The iSBC 941 Industrial Digital Processor device architecture includes on-chip CPU, data memory space, program storage space, an event timer, a status register, and 16 I/O lines (figure 1-1). Eight data lines and four control lines are used to interface the iSBC 941 IDP to the host CPU.

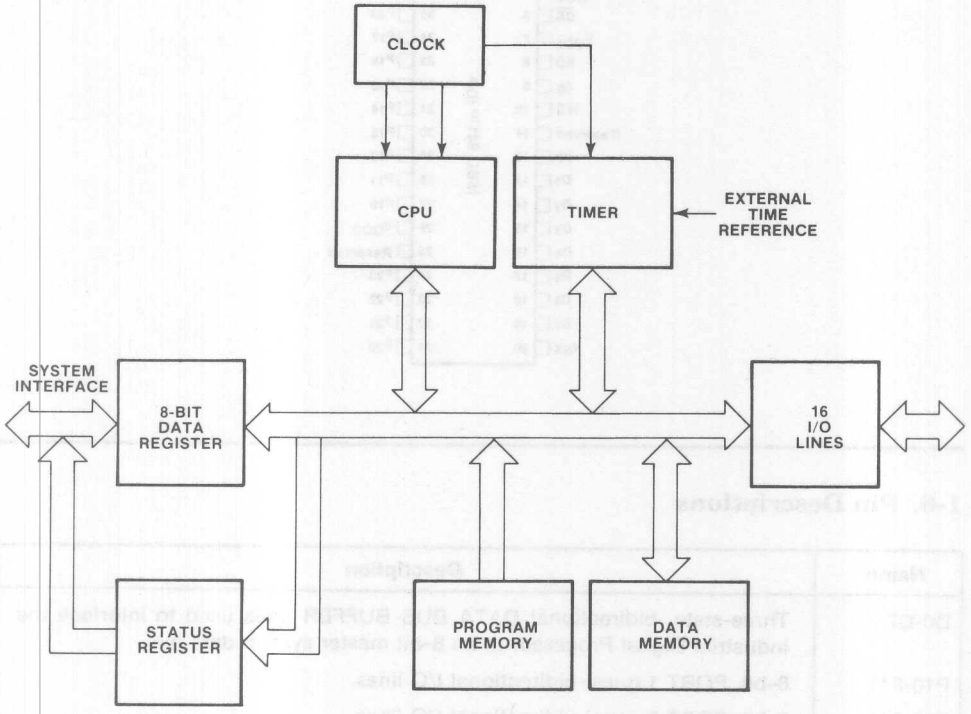


Figure 1-1. iSBC 941 IDP Block Diagram

1-3. DOCUMENTATION

In addition to this manual, the following document is available through the Intel Literature Department:

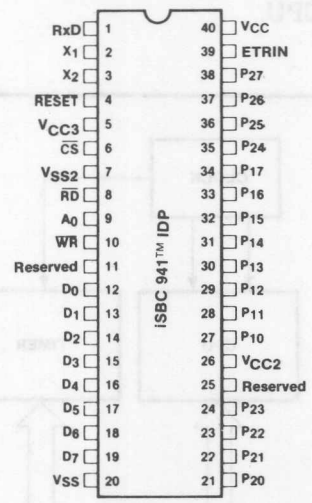
Intel Application Note: *Closed Loop Control Using the Intel iSBC 569/941 Processors, AP-60.*

1-4. SPECIFICATIONS

The following specifications are provided in this section:

1. Pin-out diagram;
2. Pin descriptions;
3. Absolute maximum ratings;
4. D.C. and operating characteristics;
5. A.C. characteristics;
6. Read and Write waveforms.

1-5. Pin-Out Diagram



1-6. Pin Descriptions

Name	Description
D0-D7	Three-state, bidirectional DATA BUS BUFFER lines used to interface the Industrial Digital Processor to an 8-bit master system data bus.
P10-P17	8-bit, PORT 1 quasi-bidirectional I/O lines.
P20-P27	8-bit, PORT 2 quasi-bidirectional I/O lines.
WR/	I/O write input which enables the host CPU to write data and commands to the INPUT DATA BUS BUFFER.

Name	Description (Cont'd)
RD/	I/O read input which enables the host CPU to read data and status words from the OUTPUT DATA BUS BUFFER or status register.
CS/	Chip select input used to select one device out of several connected to a common data bus.
A0	Address input used by the host processor to indicate whether byte transfer is data or command.
RxD	Data input pin for SERIN primary functions.
X1, X2	Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
RESET/	Input used to reset status flip-flops and to set the program counter to zero.
V_{CC}, VCC2, V_{CC3}	+5V power supply pin.
V_{DD}	+5V during normal operation.
V_{SS}, VSS2	Circuit ground potential.
ETRIN	External reference input pin.

1-7. Absolute Maximum Ratings*

Ambient Temperature Under Bias ..... 0°C to 70°C
Storage Temperature ..... -65°C to +150°C
Voltage on Any Pin With Respect to Ground ..... 0.5V to +7V
Power Dissipation ..... 1.5 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

1-8. D.C. and Operating Characteristics

$T_A = 0°C$ to $70°C$, $V_{SS} = 0V$, $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage (All Except X_1, X_2)	-0.5	0.8	V	
V_{IH1}	Input High Voltage (All Except X_1, X_2, $\overline{RESET}$)	2.2	V_{CC}		
V_{IH2}	Input High Voltage (X_1, X_2, $\overline{RESET}$)	3.0	V_{CC}	V	
V_{OL1}	Output Low Voltage (D_2-D_7)		0.45	V	$I_{OL} = 2.0$ mA
V_{OL2}	Output Low Voltage (All Other Outputs)		0.45	V	$I_{OL} = 1.6$ mA
V_{OH1}	Output High Voltage (D_0-D_7)	2.4		V	$I_{OH} = -400\ \mu A$
V_{OH2}	Output High Voltage (All Other Outputs)	2.4		V	$I_{OH} = -50\ \mu A$
I_{IL}	Input Leakage Current (RxD, ETRIN, RD, WR, CS, A_0, EA)		± 10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{OZ}	Output Leakage Current (D_0-D_7, High Z State)		± 10	μA	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$
I_{LI1}	Low Input Load Current ($P_{10}P_{17}$, $P_{20}P_{27}$		0.5	mA	$V_{IL} = 0.8V$
I_{LI2}	Low Input Load Current ($\overline{RESET}$)		0.2	mA	$V_{IL} = 0.8V$
I_{DD}	V_{DD} Supply Current		1.5	mA	
$I_{CC} + I_{DD}$	Total Supply Current		125	mA	

1-9. A.C. Characteristics

$T_A = 0°C$ to $70°C$, $V_{SS} = 0V$, $V_{CC} = V_{DD} = +5V \pm 5\%$

DBB READ

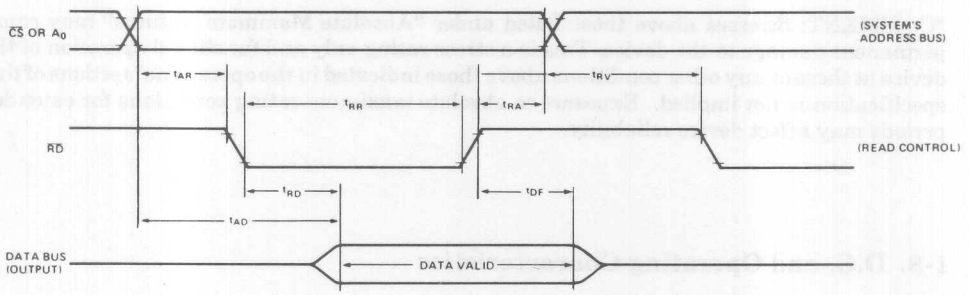
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}\downarrow$	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}\uparrow$	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	$t_{CY} = 2.5\ \mu s$
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		225	ns	$C_L = 150$ pF
t_{RD}	$\overline{RD}\downarrow$ to Data Out Delay		225	ns	$C_L = 150$ pF
t_{RDF}	$\overline{RD}\uparrow$ to Data Float Delay		100	ns	
t_{RV}	Recovery Time Between Reads And/Or Write	300		ns	
t_{CY}	Cycle Time	2.5	15	μs	

DBB WRITE

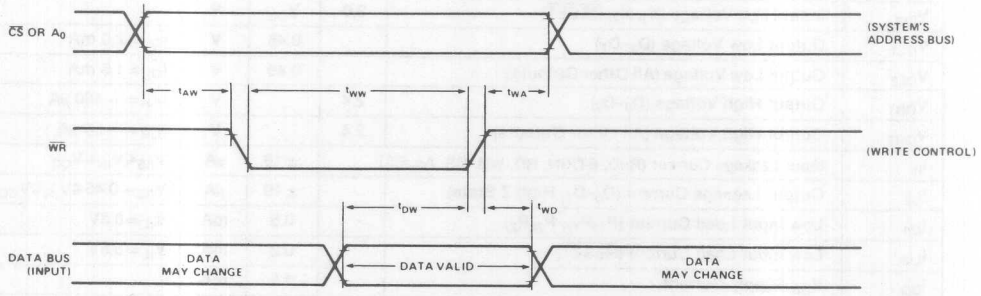
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	CS, A_0 Setup to $\overline{WR}\downarrow$	0		ns	
t_{WA}	CS, A_0 Hold After $\overline{WR}\uparrow$	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}\uparrow$	150		ns	
t_{WD}	Data Hold Aftert $\overline{WR}\uparrow$	0		ns	

1-10. Waveforms

1. READ OPERATION—DATA BUS BUFFER REGISTER.



2. WRITE OPERATION—DATA BUS BUFFER REGISTER.



CHAPTER 2
INTERFACING INFORMATION

2-1. INTRODUCTION

This chapter provides basic information describing how the iSBC 941 Industrial Digital Processor should be interfaced with the host processor. Also included is a short description of each command and primary function.

Refer to Appendix F if the iSBC 941 IDP is installed on either the iSBC 80/30 Single Board Computer or the iSBC 569 Intelligent Digital Controller.

2-2. HARDWARE INTERFACE REQUIREMENTS

The iSBC 941 IDP is interfaced as a peripheral to a host CPU. This scheme requires eight data lines (DB0-DB7) and four control lines:

1. CS/: Chip select. Selects the IDP device. Allows bussing of RD/ and WR/ to all peripherals.
2. A0: Address line used to indicate whether the byte transfer is data or command.
3. WR/: I/O write input which enables the host CPU to write data and command words to the input data bus buffer.
4. RD/: I/O read input which enables the host CPU to read data and status words from the output data bus buffer or status register.

A signal mnemonic which ends with a slash (e.g., CS/) indicates it is active low. Conversely, a signal mnemonic without a slash (e.g., A0) is active high.

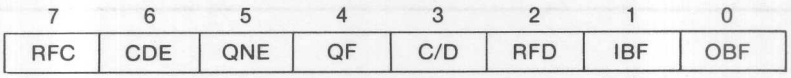
Table 2-1 illustrates the conditions necessary for the possible transfers.

Table 2-1. Transfer Table

A0	CS/	RD/	WR/	Description
0	0	0	1	Host CPU reads the IDP data output register
0	0	1	0	Host CPU writes to the IDP data input register
1	0	0	1	Host CPU reads the IDP status register
1	0	1	0	Host CPU writes to the IDP command register

The status register is an internal IDP register used for various primary functions and commands. The register is shown in figure 2-1. Register operation is described in section 3-3.

When the host CPU writes to the IDP, data bits DB0-DB7 are latched into the IDP Command/Data input register; A0 is latched into the C/D flag and the input buffer full flag is set (IBF=1). When the IDP reads the contents of the Command/Data input register, the IBF flag is cleared (IBF=0).



7	6	5	4	3	2	1	0
RFC	CDE	QNE	QF	C/D	RFD	IBF	OBF

OBF	Output Buffer Full (1=Full)
IBF	Input Buffer Full (1=Full)
RFD	Ready For Data (1=Ready)
C/D	Command/Data flag (1=Command 0=Data)
QF	Queue Full (1=Full)
QNE	Queue Not Empty (1=Not Empty)
CDE	Command/Data Error (1=Error)
RFC	Ready For Command (1=Ready)

Figure 2-1. Hardware Status Register

2-3. SOFTWARE INTERFACE REQUIREMENTS

Since the iSBC 941 IDP is a slave processor to a host CPU, the software interface requirements are relatively simple. The host CPU may issue a command request to the iSBC 941 IDP at any time. If another request had already been issued, and the IDP was currently processing that request, the current cycle would finish, and the new command would begin its processing.

2-4. BASIC OPERATION

The iSBC 941 IDP may operate at three different levels. The first level consists of basic I/O commands, such as set and clear selective bits on either of the two 8-bit I/O ports. The second level is realized by activating a specific I/O algorithm, or primary function. This will involve reading or writing to perform a predefined task by the use of a special subroutine residing in code. The third level of operation is a combination of the first two levels. In this mode the primary function is performed, interleaved with basic I/O commands. This allows maximum usage of I/O lines not required by a primary function.

2-5. COMMAND OVERVIEW

The iSBC 941 IDP commands may be divided into two broad categories:

1. Utility commands;
2. Control commands.

Simply stated, utility commands are general purpose commands, usually associated with moving a byte of information from one place to another, or reading the status of the iSBC 941 IDP. Control commands are used to control operation of the primary functions. Certain utility commands are used *only* in conjunction with a particular primary function.

Table 2-2 summarizes the iSBC 941 IDP utility commands. Table 2-3 summarizes the iSBC 941 IDP control commands. Complete descriptions of these commands are provided in Chapter 3. Complete descriptions of these commands are provided in Chapter 3.

The nine primary functions are listed in table 2-4. The first five functions are *input* oriented, while the last four functions are *output* oriented. Complete descriptions of the primary functions are provided in Chapter 3.

Table 2-2. Utility Commands

Mnemonic	Description
IDEN	Requests the identity code of the iSBC 941 IDP.
RDP1	Reads Port 1 inputs on IDP and sends data to host CPU.
WRP1	Writes data from host CPU to IDP Port 1 outputs.
ENP1IN	Indicates which Port 1 lines are being used for general purpose input.
SETP1	Sets selected IDP Port 1 lines.
CLRP1	Clears selected IDP Port 1 lines.
RDP2	Reads Port 2 inputs of IDP and sends data to host CPU.
WRP2	Writes data from host CPU to IDP Port 2 outputs.
ENP2IN	Indicates which Port 2 lines are being used for general purpose input.
SETP2	Sets selected IDP Port 2 lines.
CLRP2	Clears selected IDP Port 2 lines.
RDFQ	Reads next byte from the IDP FIFO queue.
RDIDV	Reads the IDV status flags used by the PERIOD primary function.
SETSF	Sets the scale factor value for determining actual TRP.
SETOE	Enables or disables the Port 2 outputs used by the primary functions.
SETEC0-SETEC7	Set EVENT counters (time-of-day)
RDEC0-RDEC7	Reads the event accumulators used by the EVENT primary function.
LATCH	Transfers all eight counter values used by the EVENT primary function to latch registers.
RDLC0-RDLC7	Reads the latched registers used by EVENT.
RDPC0-RDPC3 (L&H)	Read measurement counters used by PERIOD Primary Function.
RDFC0-RDFC7 (L&H)	Read measurement counters used by FCOUNT Primary Function
SETGP	Sets the SHOT1 output lines to be active low or high.
S20DLY-S27DLY	Sets the SHOT 1 delay parameters.
S20PRD-S27PRD	Sets the SHOT1 period parameters.
F20DLY-F27DLY	Sets the FREQ delay parameters.
F20PRD-F27PRD	Sets the FREQ period parameters.

Table 2-3. Control Commands

Mnemonic	Description
PACIFY	Software reset. All I/O lines revert to the input state, and all control variables are cleared. Does not clear FLAG mode (see section 3-6).
INITPF	Selects the desired primary function; initializes the parameters used by a primary function.
LOOP	Executes the selected primary function in a cyclical manner.
PAUSE	Exits LOOP or INITPF mode; resets the IDP error flag in the status register.

Table 2-4. Primary Functions

Mnemonic	Description
SCAN	Monitors up to 16 input lines for a change of state.
EVENT	Monitors up to eight input lines for event sensing; also can be used as a programmable divider.
PERIOD	Measures the period of up to four independent inputs.
FCOUNT	Measures the frequency of up to eight independent inputs.
SERIN	Provides for simplex asynchronous serial input of data characters.
SEROUT	Provides for simplex asynchronous serial output of data characters.
SHOT1	Emulates a gated one-shot pulse generator circuit, on up to eight lines.
FREQ	Generates up to eight frequency outputs with programmable pulse width and period.
STEPPER	Generates up to eight independent outputs suitable for driving stepper motors.

CHAPTER 3
SYSTEM OPERATION

3-1. INTRODUCTION

This chapter provides expanded command descriptions, detailed primary function descriptions, and a discussion of overall system operation. Programming examples are provided for initialization sequences. The information in this chapter provides the framework to set up the IDP for a specific application.

3-2. PERIPHERAL I/O MODE

The iSBC 941 IDP can be used to perform simple bit-set and bit-clear I/O operations. This is done by using any of the following commands:

WRP1, WRP2: Write to port 1, 2.

RDP1, RDP2: Read from ports 1, 2.

SETP1, SETP2: Set ports 1, 2.

CLRP1, CLRP2: Clear ports 1,2.

The host CPU can issue commands at any time. Hex codes for these commands are listed in Appendix A.

3-3. INTELLIGENT I/O PROCESSOR MODE

The intelligent I/O processor mode is utilized when a primary function is requested and activated. Primary functions are pre-defined I/O algorithms, each with a number of programmable parameters which suit a variety of applications. Although there are nine primary functions available, only one may be used at a time. The desired primary function is selected and initialized by the INITPF command, and activated by the LOOP command. If a new primary function is desired while another is processing, the PAUSE control command is requested, stopping the current function. The new primary function is then requested with the INITPF command, and activated by the next LOOP command.

The INITPF command will initialize all of the parameters required by the particular primary function. The first parameter always selects which function of the nine is called. The remaining parameters are function dependent. The PAUSE command is always used to terminate the initialization sequence. The PAUSE command can be requested at any time, even if all of the primary function parameters have not been explicitly initialized; this allows the host CPU to send only those parameters required by the application.

All of the primary functions, except FCOUNT, are executed at a controlled rate called the time reference period (TRP). Each primary function has a minimum TRP required for proper operation. This minimum TRP must be adjusted if background processing is desired (section 3-4). Two different clocks may be used to obtain the TRP: an external clock or an internal clock. A scale factor parameter byte allows the TRP to be adjusted over an eight bit range (section 3-7).

3-4. BACKGROUND PROCESSING

It is possible to request simple I/O utility commands while the iSBC 941 IDP is executing one of the primary functions. These commands will be processed when the

primary function has finished its LOOP cycle, and before the next TRP begins. Typically, this mode will be efficient when a particular primary function is servicing only a few of the possible input or output lines. In maximum configurations, where all lines are being utilized by the primary functions, a longer TRP will be required to allow for background processing.

If the selected primary function does not utilize all of the 16 I/O lines, the unused lines are available for background processing. For example, if using the STEPPER primary function, all Port 1 lines are available for background processing. Read, write, set and clear utility commands could be requested for Port 1, while the STEPPER function is operating. Note that the minimum TRP must be increased to allow ample time for the background processing task to be performed (refer to Appendices A and B for utility command and primary function execution times).

3-5. UTILITY COMMANDS

The iSBC 941 IDP has numerous utility commands which can be used to examine and modify the internal registers and all 16 I/O lines. Each utility command may be requested when the iSBC 941 IDP is idle, or while LOOPing a primary function. If a utility command is requested while a primary function is LOOPing, the utility command execution time (refer to Appendix A) must be added to the minimum TRP required by the selected primary function (refer to Appendix B).

CAUTION

If mixing inputs and outputs on a single port, always write a one (1) to the lines used as inputs, before executing a primary function.

Due to the quasi-bidirectional nature of the iSBC 941 I/O lines, the inputs and outputs can be mixed in any combination on either port. However, the host program must guarantee that any port output command always writes a logic '1' to any line that is used as an input line, or else that line can no longer be used as an input. In addition, physical damage may result due to excessive input current. The ENP1IN and ENP2IN commands are used to inform the CLRP1, SETP1, CLRP2, SETP2 utility commands and primary functions which lines are used as inputs. Once the ENP1IN command is executed, the IDP will always write a '1' to the selected lines (i.e., inputs) when a future SETP1 or CLRP1 command is requested. ENP2IN performs the same function for Port 2.

The following paragraphs describe the iSBC 941 utility commands. Command codes are listed in Appendix A.

1. IDENTIFY IDP TYPE (IDEN)

The IDEN command is used to request the identity code (ID) of the IDP. The iSBC 941 IDP ID is 41 (Hex).

2. READ PORT 1 (RDP1)

RDP1 is used by the host CPU to directly read the current port 1 inputs on the IDP. This operation will not disturb any lines designated as outputs. RDP1 returns a single parameter.

3. WRITE PATTERN TO PORT 1 (WRP1)

WRP1 can be used by the host CPU to output a data pattern directly to port 1 on the IDP. This pattern will be latched by the IDP. WRP1 requires a single parameter.

4. SET PORT 1 INPUT MASK (ENP1IN)

ENP1IN informs the iSBC 941 IDP which Port 1 lines are being used as general-purpose inputs. ENP1IN requires a single parameter byte. This mask is used by SETP1, CLRP1, and LOOP, preventing these functions from 'clearing' those inputs.

5. SET BITS OF PORT 1 (SETP1)

The SETP1 utility command sets a single bit or a group of bits on Port 1. SETP1 requires a single parameter which will be logically-ORed to Port 1. Any bits in the parameter which are set will cause the corresponding bits of Port 1 to be set; SETP1 will also write a 1 to any bit which is designated as an input (see ENP1IN); all other Port 1 lines will remain unchanged.

6. CLEAR BITS OF PORT 1 (CLRP1)

The CLRP1 utility command clears a single bit or a group of bits on Port 1. CLRP1 requires a single parameter byte which will be complemented and then logically-ANDed with Port 1. Any bits in the parameter which are set will cause the corresponding bits of Port 1 to be cleared; CLRP1 will also write a 1 to any bit which is designated as an input (see ENP1IN); all other Port 1 lines will remain unchanged.

7. READ PORT 2 (RDP2)

RDP2 is identical to RDP1, except the current input of port 2 is returned to the host CPU. RDP2 returns a single parameter.

8. WRITE PATTERN TO PORT 2 (WRP2)

WRP2 is identical to WRP1, except the output pattern is sent to port 2. WRP2 requires a single parameter.

9. SET PORT 2 INPUT MASK (ENP2IN)

ENP2IN informs the iSBC 941 IDP which Port 2 lines are being used as general-purpose inputs. ENP2IN requires a single parameter byte. This mask is used by SETP2, CLRP2, STEP, and LOOP, preventing these functions from 'clearing' those inputs.

CAUTION

If inputs and outputs are mixed on either Port 1 or Port 2, ensure that input impedance does not draw excessive current from the IDP.

10. SET BITS OF PORT 2 (SETP2)

The SETP2 utility command sets a single bit or a group of bits on Port 2. SETP2 requires a single parameter byte which will be logically-ORed to Port 2. Any bits in the parameter which are set will cause the corresponding bits of Port 2 to be set; SETP2 will also write a 1 to any bit which is designated as an input (see ENP2IN); all other Port 2 lines will remain unchanged.

11. CLEAR BITS OF PORT 2 (CLRP2)

The CLRP2 utility command clears a single bit or a group of bits on Port 2. CLRP2 requires a single parameter byte which will be complemented and then logically-ANDed with Port 2. Any bits in the parameter which are set will cause the corresponding bits of Port 2 to be cleared; CLRP2 will also write a 1 to any bit which is designated as an input (see ENP2IN); all other Port 2 lines will remain unchanged.

12. READ FIFO QUEUE (RDFQ)

This command reads messages which are stored in a FIFO queue by the SCAN primary function. Each time the RDFQ command is requested by the host CPU, the next message is read. Queue management is accomplished with the iSBC 941 IDP hardware status register. Refer to section 3-9, SCAN primary function, for a description of register operation.

13. READ 'INPUT DATA VALID' FLAGS (RDIDV)

The RDIDV utility command is used to read the IDV status flags used by the FCOUNT and PERIOD primary functions. These status flags are used to indicate when the measurement registers contain valid data. The IDV flags are automatically cleared when the INITPF command selects a primary function. When the IDV command is invoked by the host CPU, all eight IDV flags will be loaded into the output register, setting the OBF flag. If the OBF flag is already set when the RDIDV command is requested, the 'Command/Data Error' (CDE) flag is set in the hardware status register. RDIDV requires no parameters.

14. SET SCALE FACTOR (SETSF)

SETSF allows the host CPU to set the scale factor used by the primary functions. The host CPU can set the desired scale factor from 1 to 256. The minimum Time Reference Period (TRP) that can be used by a primary function is listed in Appendix B. SETSF requires a single parameter byte, which is the 2's complement of the desired scale factor. (Section 3-7.)

15. SET PRIMARY FUNCTION OUTPUT ENABLE PARAMETER (SETOE)

SETOE is used to enable/disable the Port 2 outputs used by the primary functions. The output enable parameter can be sent to the IDP with either the INITPF command or the SETOE command. The SETOE command can be used to modify the Output Enable parameter while the iSBC 941 IDP is executing a primary function, without disturbing the remaining outputs. SETOE requires a single parameter byte.

The following utility commands are used only with a specific primary function, as indicated by the title and description.

16. SET EVENT COUNTER 0-7 (SETECO-SETEC7)

These commands allow setting a preselected count into the EVENT counters. There is one command for each input counter.

17. READ EVENT COUNTERS 0-7 (RDEC0-RDEC7)

The RDEC commands are used to read the event accumulators used by the EVENT primary function. There is a separate RDEC command for each accumulator. The RDEC commands require no parameters. If EVENT is programmed for cascaded counters, the LATCH and RDLC commands should be used instead. (See 18, 19.)

18. LATCH EVENT COUNTERS (LATCH)

LATCH allows the host CPU to transfer all eight counters used by the EVENT primary function to a separate set of latches. The RDLC command can be used to read the latched counts. LATCH requires no parameters.

19. READ LATCHED COUNT 0-7 (RDLC0-RDLC7)

The RDLC commands are used to read the latched event counters. There is a

separate RDLC command for each event counter. The RDLC commands require no parameters.

20. READ PERIOD COUNTER 0-3 LOW BYTE (RDPC0L-RDPC3L)

This command is used with the PERIOD primary function to read the lower byte of each 16-bit measurement counter. There is a separate command and counter for inputs P10-P13.

21. READ PERIOD COUNTER 0-3 HIGH BYTE (RDPC0H-RDPC3H)

This command is used with the PERIOD primary function to read the upper byte of each 16-bit measurement counter. There is a separate command and counter for inputs P10-P13.

22. READ FCOUNT COUNTER 0-7 LOW BYTE (RDFC0L-RDFC7L)

These commands are used by FCOUNT to read the lower byte of each 16-bit measurement counter. There is a separate command and counter for each input line, P10 through P17.

23. READ FCOUNT COUNTER 0-7 HIGH BYTE (RDFC0H-RDFC7H)

These commands are used by FCOUNT to read the upper byte of each 16-bit measurement counters. There is a separate command and counter for each input line, P10 through P17.

24. SET SHOT1 GATE POLARITY (SETGP)

This command sets each output line used by the SHOT1 primary function to either active low (1) or active high (0). Bit 0 corresponds to output line P20.

25. SET SHOT1 DELAY PARAMETERS (S20DLY-S27DLY)

These commands are used to set the delay parameter of the individual output lines used by the SHOT1 primary function. There is one command for each output.

26. SET SHOT1 PERIOD PARAMETERS (S20PRD-S27PRD)

These commands are used to set the period parameter of the individual output lines used by the SHOT1 primary function. There is one command for each output.

27. SET FREQ DELAY PARAMETERS (F20DLY-F27PRD)

Same as S20DLY-S27DLY, except used with FREQ primary function.

28. SET FREQ PERIOD PARAMETERS (F20PRD-F27PRD)

Same as S20PRD-S27PRD, except used with the FREQ primary function.

3-6. CONTROL COMMAND DESCRIPTIONS

The following control commands are used by the iSBC 941 IDP. The code for each command is listed in Appendix A.

1. ENABLE SPECIAL INTERRUPT MODE (ENFLAG)

Instead of polling the IDP hardware status register, it is possible to operate the IDP in an interrupt-driven environment. The ENFLAG command forces the IDP to generate interrupt outputs based on the IBF and OBF status flags. When the

ENFLAG command is requested, the IDP will change the function of I/O lines P24 and P25. If OBF=1, then P24 will also be 1. If OBF=0, then P24 will also be 0. If IBF=0, then P25 will be 1; if IBF=1 then P25 will be a 0. Thus, P24 indicates that the IDP has data ready for the host CPU in its output register. Likewise, P25 indicates that the IDP input register is available (i.e., the IDP has read the last command or parameter byte).

The ENFLAG mode can be cleared only with a hardware reset. However, P24 and P25 can be individually disabled or enabled by the CLRP2 and SETP2 commands.

2. SOFTWARE RESET (PACIFY)

The PACIFY command is used to reset the IDP as close as possible to the same initial condition as a hardware reset. PACIFY will reset all I/O lines to the input state, clear of all the IDP registers, and initialize any control variables required by the iSBC 941 IDP. After either a software or a hardware reset, the host program must re-select a primary function with the INITPF command. PACIFY cannot reset the FLAG mode or clear the Output Buffer.

3. SELECT AND INITIALIZE A PRIMARY FUNCTION (INITPF)

This command is used to select one of the nine primary functions, and initialize all of the parameters required by that particular function. The initialization sequence is always terminated by the PAUSE command after the host CPU has sent the last parameter required by the application.

The INITPF command is normally used to initialize the parameters required by a specific primary function, since the host CPU only has to send one command byte followed by the parameters. However, if the application requires parameter modification while LOOPing, INITPF cannot be used, since INITPF always stops the LOOP mode. Utility commands 13-16 and 24-28 can be used to change individual parameters without exiting the LOOP mode.

4. EXECUTE PRIMARY FUNCTION (LOOP)

The LOOP command forces the IDP to execute the selected primary function at a controlled rate (section 3-7), without further intervention from the host CPU. When LOOP is requested, the "ready for command" (RFC) flag in the hardware status register is cleared, and will remain so until:

1. An unrecoverable error is detected (e.g., queue overflow).
2. The PACIFY command is requested.
3. The PAUSE command is requested.
4. All inputs have been measured (FCOUNT primary function only).

If any of these conditions occur the iSBC 941 IDP will exit the LOOP mode, set the RFC flag, and wait for the next command from the host CPU.

5. HALT PRIMARY FUNCTION (PAUSE)

The PAUSE command can be used to terminate the primary function initialization sequence. It can also be used to force the IDP to exit the LOOP mode (e.g., to halt the current primary function, in order to initialize another primary function). The PAUSE command will not change any of the I/O lines or registers, except the "command/data error" (CDE) flag in the IDP status register.

3-7. TIME REFERENCE PERIOD

When the LOOP command is requested, the selected PF is executed within a specific

time period called the "time reference period" or simply, TRP. The TRP is a direct function of the fixed clock rate which is used by the iSBC 941 IDP and the scale factor parameter specified during primary function initialization (section 3-8).

Each primary function (except FCOUNT) must not exceed a *minimum* TRP to ensure proper operation. These minimum TRP rates are listed in Appendix B.

The scale factor parameter byte should be the 2's complement of the SF value desired in the equations below.

Two TRP clock sources are available to the iSBC 941 IDP: an internal (on-chip—clock or an external reference source (connected to the ETRIN input). The clock source is selected by bit 7 of the select PF parameter, where 1=external and 0=internal. If the internal timer is used, the TRP is calculated as follows:

$$TRP = T_{cy} \times 32 \times SF$$

where T_{cy} is the IDP execution speed, and SF is the selected scale factor value.

If the external reference is selected, the TRP is calculated as follows:

$$TRP = T_{ex} \times SF$$

where T_{ex} is the external reference period, and SF is the selected scale factor value.

3-8. INITIALIZING A PRIMARY FUNCTION

All primary functions are initialized in the same manner. The host CPU must first request the initializing command (INITPF). This byte will be followed by several parameter bytes depending on the particular primary function selected and number of port lines being used (refer to figure 3-2). Notice that the first parameter byte will always be the select byte, which specifies the primary function being selected, and also indicates which TRP reference is to be used. A list of parameters for each of the primary functions is provided in section 3-9, along with a description of the function.

After all the parameters have been sent from the host CPU to the iSBC 941 IDP, the PAUSE command must be requested to terminate the initialization sequence.

The host CPU needs send only those parameters which the primary function will utilize for that particular application. Table 3-1 is an example of how to initialize a primary function.

Table 3-1. Initialization Sequence

```
/* INITIALIZE PRIMARY FUNCTION */
PF$INIT: PROCEDURE (TBL$PTR,TBL$SIZ) PUBLIC;
   DECLARE TBL$PTR ADDRESS, TBL$SIZ BYTE;
   DECLARE I BYTE;
   DECLARE PARAM BASED TBL$PTR BYTE;
   CALL SND$CMD (INITPF);
   DO I = 0 TO (TBL$SIZ - 1);
      CALL SND$DAT (PARAM);
      TBL$PTR=TBL$PTR+1;
   END
   CALL SND$CMD (PAUSE);
END PF$INIT;
```

A new primary function may be requested by simply issuing another INIT command and the required parameter bytes. To begin operation, the LOOP command must be requested.

3-9. PRIMARY FUNCTION DESCRIPTIONS

The following paragraphs describe the primary functions which operate in the input mode. Each description gives a list of parameters for each primary function, and a set of sample declarations for the utility commands used by the particular primary function. A sample PL/M 80 data structure is provided, showing how to initialize the parameters. Refer to Appendices D and E, for additional programming information; refer to Appendices A, B, and C for command codes and timing information.

1. SCAN FOR CHANGE OF STATE (SCAN)

The SCAN primary function can be used to monitor 16 input lines for a change of state, using Port 1 and Port 2 (figure 3-1).

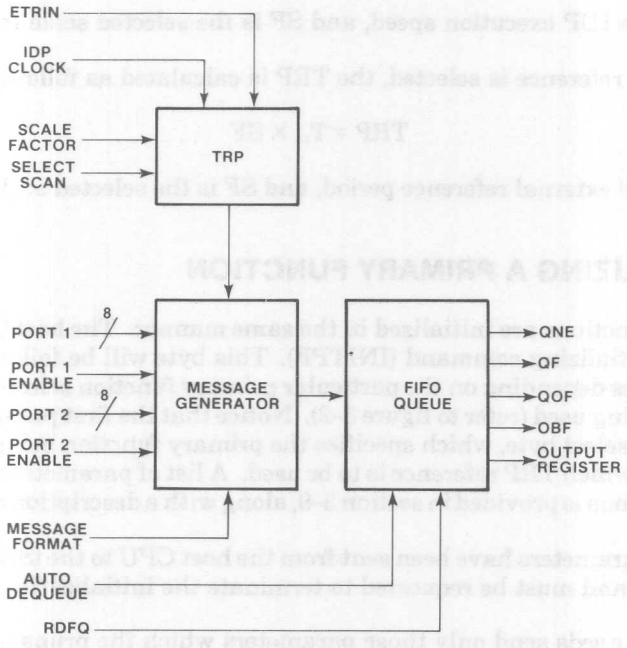


Figure 3-1. SCAN Functional Block Diagram

To select the SCAN primary function, the following parameters must be sent to the IDP (table 3-2):

Table 3-2. SCAN Data Structure

```
/* DATA STRUCTURE USED BY 'SCAN' PRIMARY FUNCTION */
DECLARE SCAN STRUCTURE (
   SELECT BYTE,
   SCALE$FACTOR BYTE,
   MESSAGE$FORMAT BYTE,
   PORT$1$ENABLE BYTE,
   PORT$2$ENABLE BYTE );

/* UTILITY COMMANDS USED BY SCAN PRIMARY FUNCTION
DECLARE RDFQ LITERALLY '0FH'; /* READ BYTE FROM QUEUE */
```

SCAN select byte: selects the SCAN function and indicates the time reference source with bit 7.

Scale factor byte: sets the duration of the TRP (section 3-7).

Message format byte: indicates the number of bytes which will comprise the message byte. Messages may contain 1, 2, 3, or 4 bytes.

Port 1 enable byte: enables the selected Port 1 lines (1=enable; 0=disable).

Port 2 enable byte: enables the selected Port 2 lines (1=enable; 0=disable).

When SCAN is activated by the LOOP command, the Port 1 and Port 2 inputs will be sampled once every TRP. If any line has changed since the last TRP and that input has its port enable bit set, a SCAN message will be generated. Depending on the message format parameter, the message will indicate:

a. Which Port 1 inputs have changed.
b. The current state of Port 1.
c. Which Port 2 inputs have changed.
d. The current state of Port 2.

Only one message can be generated each TRP, and then only if an enabled input has changed. All messages will have the same number of bytes, either 1, 2, 3, or 4 depending on the message format parameter selected during initialization.

Byte A	Port 1	Change State
Byte B	Port 1	Present State
Byte C	Port 2	Change State
Byte D	Port 2	Present State

Sample Message

The messages are buffered in a FIFO queue. The status of this queue can be indicated by the hardware status register:

B7	B6	B5	B4	B3	B2	B1	B0
—	QOF	QNE	QF	—	—	—	OBF

QOF	Queue Overflow (1=Overflow)
QNE	Queue Not Empty (1=Not Empty)
QF	Queue Full (1=Full)
OBF	Output Buffer Full (1=Full)

The host CPU can read the message queue by enabling the Auto-Dequeue mode (bit 6=1 in the select SCAN parameter byte) or by requesting the RDFQ utility command (section 3-5). If the Auto-Dequeue mode is selected, the SCAN function will monitor the queue status and the "output buffer full" (OBF) flag of the status register. If QNE=1 and OBF=0 then SCAN will load the next byte from the message queue into the IDP output register, thereby setting OBF=1. When the host CPU reads the IDP output register, the OBF flag is cleared, permitting SCAN to load the next byte into the output register.

The Auto-Dequeue mode *cannot* be used if the application requires utility commands to return data to the host CPU while SCAN is operating. The host CPU would not be able to distinguish between data from the utility command and message bytes. Queue status can still be monitored, and if QNE=1 and OBF=0, the RDFQ utility command can be requested to read the next message byte.

2. *EVENT COUNTER (EVENT)*

The EVENT primary function monitors eight input lines for pulse or event inputs (figure 3-2).

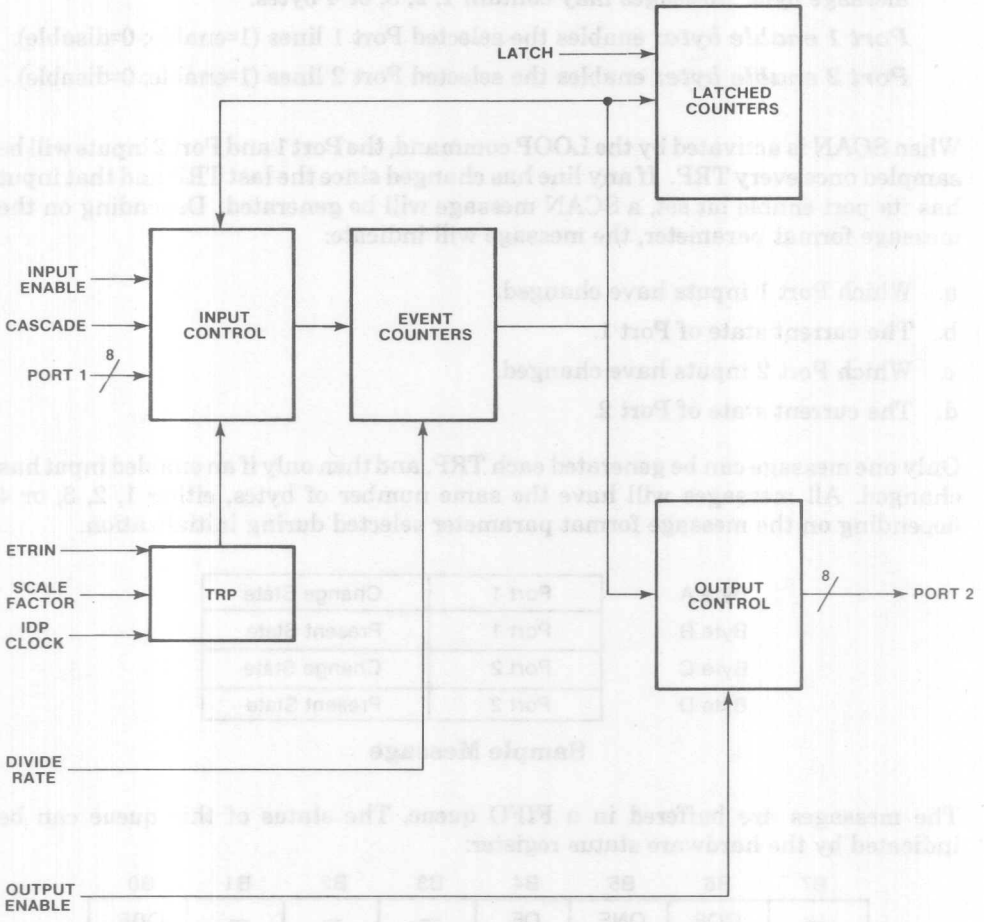


Figure 3-2. EVENT Functional Block Diagram

Each counter can be programmed as an event counter or a programmable divider. Associated with each counter is an output which toggles at ½ the divide rate. If necessary, individual event counters, which are eight bits in length, may be cascaded to provide longer counts.

To select the EVENT primary function, the following parameter bytes must be sent to the IDP (table 3-3):

Table 3-3. EVENT Data Structure

```
/* DATA STRUCTURE FOR 'EVENT' PRIMARY FUNCTION */
DECLARE VENT STRUCTURE
    SELECT BYTE,
    SCALE$FACTOR BYTE,
    OUTPUT$ENABLE BYTE,
    INITIAL$OUTPUT BYTE,
    INPUT$ENABLE BYTE,
    CASCADE BYTE,
    P10$DIVIDE$RATE BYTE,
    P11$DIVIDE$RATE BYTE,
```

Table 3-3. EVENT Data Structure (Cont.)

```
            P12$DIVIDE$RATE BYTE,
            P13$DIVIDE$RATE BYTE,
            P14$DIVIDE$RATE BYTE,
            P15$DIVIDE$RATE BYTE,
            P16$DIVIDE$RATE BYTE,
            P17$DIVIDE$RATE BYTE );

/* UTILITY COMMANDS FOR EVENT PRIMARY FUNCTION */
DECLARE  LATCH    LITERALLY  '10H';  /*  LATCH       EVENT COUNTERS */
DECLARE  RDLC0    LITERALLY  '42H';  /*  READ  P10   LATCHED COUNT  */
DECLARE  RDLC1    LITERALLY  '43H';  /*  READ  P11   LATCHED COUNT  */
DECLARE  RDLC2    LITERALLY  '44H';  /*  READ  P12   LATCHED COUNT  */
DECLARE  RDLC3    LITERALLY  '45H';  /*  READ  P13   LATCHED COUNT  */
DECLARE  RDLC4    LITERALLY  '46H';  /*  READ  P14   LATCHED COUNT  */
DECLARE  RDLC5    LITERALLY  '47H';  /*  READ  P15   LATCHED COUNT  */
DECLARE  RDLC6    LITERALLY  '48H';  /*  READ  P16   LATCHED COUNT  */
DECLARE  RDLC7    LITERALLY  '49H';  /*  READ  P17   LATCHED COUNT  */
DECLARE  RDEC0    LITERALLY  '4AH';  /*  READ  P10   EVENT COUNTER  */
DECLARE  RDEC1    LITERALLY  '4BH';  /*  READ  P11   EVENT COUNTER  */
DECLARE  RDEC2    LITERALLY  '4CH';  /*  READ  P12   EVENT COUNTER  */
DECLARE  RDEC3    LITERALLY  '4DH';  /*  READ  P13   EVENT COUNTER  */
DECLARE  RDEC4    LITERALLY  '4EH';  /*  READ  P14   EVENT COUNTER  */
DECLARE  RDEC5    LITERALLY  '4FH';  /*  READ  P15   EVENT COUNTER  */
DECLARE  RDEC6    LITERALLY  '50H';  /*  READ  P16   EVENT COUNTER  */
DECLARE  RDEC7    LITERALLY  '51H';  /*  READ  P17   EVENT COUNTER  */
DECLARE  SETEC0   LITERALLY  '8AH';  /*  READ  P10   EVENT COUNTER  */
DECLARE  SETEC1   LITERALLY  '8BH';  /*  READ  P11   EVENT COUNTER  */
DECLARE  SETEC2   LITERALLY  '8CH';  /*  READ  P12   EVENT COUNTER  */
DECLARE  SETEC3   LITERALLY  '8DH';  /*  READ  P13   EVENT COUNTER  */
DECLARE  SETEC4   LITERALLY  '8EH';  /*  READ  P14   EVENT COUNTER  */
DECLARE  SETEC5   LITERALLY  '8FH';  /*  READ  P15   EVENT COUNTER  */
DECLARE  SETEC6   LITERALLY  '90H';  /*  READ  P16   EVENT COUNTER  */
DECLARE  SETEC7   LITERALLY  '91H';  /*  READ  P17   EVENT COUNTER  */
```

EVENT select byte: selects the EVENT primary function and indicates the time reference source with bit 7.

Scale factor byte: sets the duration of the TRP (section 3-7).

Output Enable byte: enables the selected port 2 outputs. (1=enable; 0=disable.)

Initial output byte: determines the initial state of each port 2 output line. (1=high initial state; 0=low initial state.)

Input enable: enables the selected event counter/divider. (1=enable; 0=disable.)

Cascade byte: selects the event counter input source. (1=cascade; 0=Port 1.) Note: counter 0 must always use Port 1.

Divide rate: programs the count at which the event counter is cleared. A divide rate of zero is used for normal event counter mode.

When EVENT is activated by the LOOP command, the inputs will be sampled once every TRP. The outputs will be updated at the end of each TRP. Each output change will occur at integer multiples of the TRP.

If an EVENT counter (EC) is enabled for a Port 1 input, it will be incremented each time the associated Port 1 input line goes high-to-low. If the new counter value is equal to the divide rate parameter, the counter is automatically cleared, and the output, if enabled, will be complemented.

If an EVENT counter is enabled for cascade input, it will be incremented each time the adjacent counter reaches its divide rate value. For example, EC3 will be incremented when EC2 is cleared upon reaching the Port 12 divide rate. Since the cascading is done internally, any cascaded EVENT counters will not use the associated Port 1 input line. These lines can be used for general purpose I/O utility command operations.

The output enable parameter allows the IDP to generate outputs with a period equal to the TRP times the associated divide rate. The output will be complemented once when the count equals divide rate/2, and again when the count equals the divide rate. This parameter also determines the output transition direction when the EVENT counter reaches the divide rate value. If the initial output is equal to one (1) the transition will be low-to-high; if it is equal to zero (0) the transition will be high-to-low.

Each counter can be programmed to overflow on any 8-bit count, by choosing the appropriate divide rate parameter. Thus, event counters can be used as programmable dividers. A divide rate of zero should be used for normal event counters.

For example, a time-of-day counter can be configured as follows:

1. Connect 60 Hz input to P10.
2. Set P10, P11, P12 divide rate to 60.
3. Set the P11, P12, P13 cascade flags.
4. Set P13 divide rate to 24.

After the host CPU has initialized EVENT, it can activate EVENT via the LOOP command. If the host CPU wishes to read the time-of-day, it merely requests the LATCH utility command followed by RDLC1 through RDLC3 utility command requests (section 3-5).

If the application does not require cascaded counters the host CPU can read the event accumulators directly via the RDEC0-RDEC7 utility commands.

The event counters should not be cascaded externally, since an overflow that occurs during one TRP will not increment the cascaded counter until the next TRP, and the host CPU can read the counters between TRP's.

3. PERIOD MEASUREMENT (PERIOD)

The PERIOD primary function measures the period of four digital inputs, each with a resolution of 16 bits. The period of each cycle is measured, but only the last cycle is stored (figure 3-3).

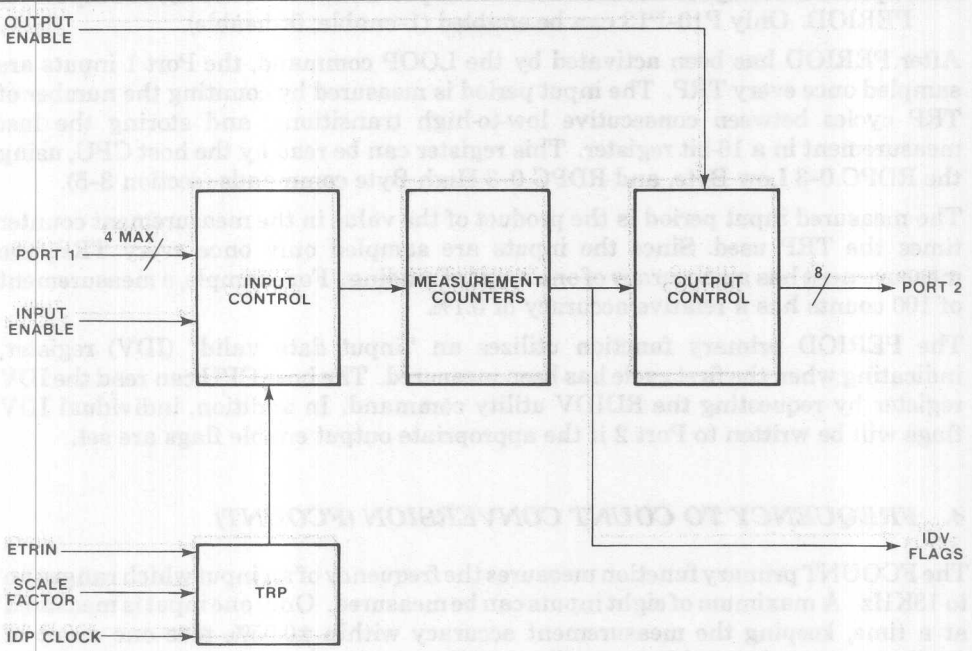


Figure 3-3. PERIOD Functional Block Diagram

The following parameters are required to operate the PERIOD primary function (table 3-4).

Table 3-4. PERIOD Data Structure

```
/* DATA STRUCTURE USED BY 'PERIOD' PRIMARY FUNCTION */
DECLARE PERIOD STRUCTURE (
   SELECT BYTE,
   SCALE$FACTOR BYTE,
   OUTPUT$ENABLE BYTE,
   INITIAL$OUTPUT BYTE,
   INPUT$ENABLE BYTE,
   NULL0 BYTE,
   NULL1 BYTE,
   NULL2 BYTE,
   NULL3 BYTE );
/* NULL0, NULL1, NULL2, NULL3 = 0; */


 /* UTILITY COMMANDS USED BY PERIOD PRIMARY FUNCTION */
DECLARE  RDPC0L  LITERALLY  '42H';  /* READ P10 LOW BYTE  */
DECLARE  RDPC0H  LITERALLY  '43H';  /* READ P10 HIGH BYTE */
DECLARE  RDPC1L  LITERALLY  '44H';  /* READ P11 LOW BYTE  */
DECLARE  RDPC1H  LITERALLY  '45H';  /* READ P11 HIGH BYTE */
DECLARE  RDPC2L  LITERALLY  '46H';  /* READ P12 LOW BYTE  */
DECLARE  RDPC2H  LITERALLY  '47H';  /* READ P12 HIGH BYTE */
DECLARE  RDPC3L  LITERALLY  '48H';  /* READ P13 LOW BYTE  */
DECLARE  RDPC3H  LITERALLY  '49H';  /* READ P13 HIGH BYTE */
```

PERIOD select byte: selects the PERIOD function and indicates the time reference source with bit 7.

Scale factor byte: sets the duration of the TRP (section 3-7).

Output enable byte: enables the status of the IDV flags to be written to Port 2. (section 3-5). Only P20-P23 can be enabled (1=enable; 0=disable).

Input enable byte: used to select which port 1 lines are to be used as inputs for PERIOD. Only P10–P13 can be enabled (1=enable; 0=disable).

After PERIOD has been activated by the LOOP command, the Port 1 inputs are sampled once every TRP. The input period is measured by counting the number of TRP cycles between consecutive low-to-high transitions, and storing the last measurement in a 16-bit register. This register can be read by the host CPU, using the RDPC 0–3 Low Byte, and RDPC 0–3 High Byte commands (section 3–5).

The measured input period is the product of the value in the measurement counter times the TRP used. Since the inputs are sampled only once every TRP, the measurement has an accuracy of one count of reading. For example, a measurement of 100 counts has a relative accuracy of 0.1%.

The PERIOD primary function utilizes an "input data valid" (IDV) register, indicating when the first cycle has been measured. The hose CPU can read the IDV register by requesting the RDIDV utility command. In addition, individual IDV flags will be written to Port 2 if the appropriate output enable flags are set.

4. FREQUENCY TO COUNT CONVERSION (FCOUNT)

The FCOUNT primary function measures the frequency of an input which ranges up to 18KHz. A maximum of eight inputs can be measured. Only one input is measured at a time, keeping the measurement accuracy within ±0.05%, plus one count of reading, over a sixteen bit range (figure 3-4).

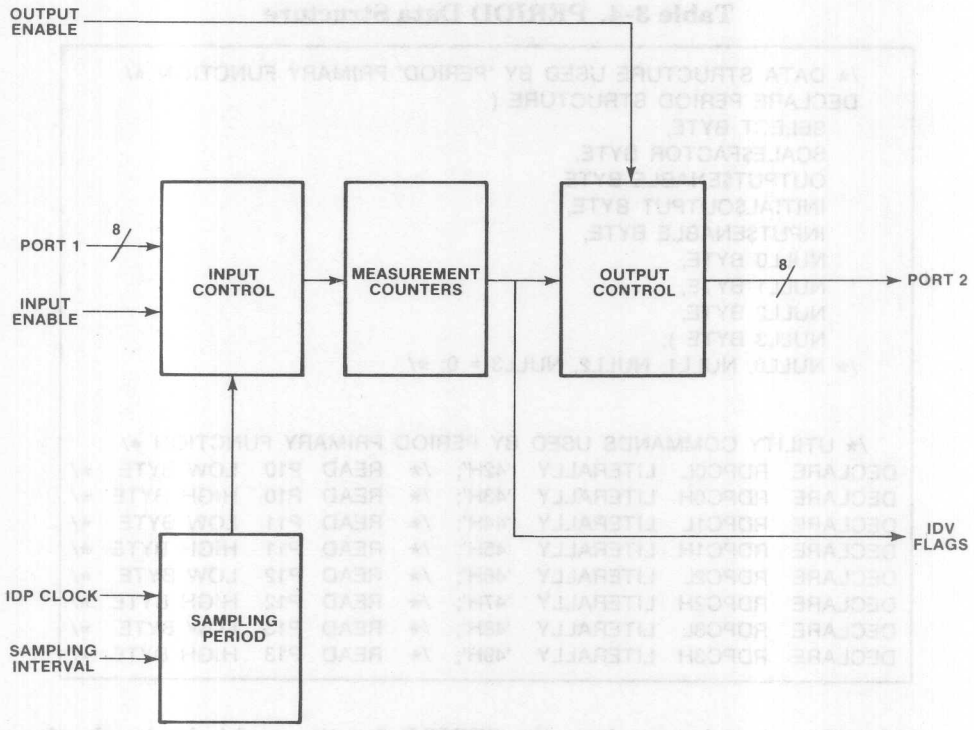


Figure 3-4. FCOUNT Functional Block Diagram

The following parameters are required to initialize FCOUNT (table 3-5):

Table 3-5. FCOUNT Data Structure

```
/* DATA STRUCTURE USED BY 'FCOUNT' PRIMARY FUNCTION */
DECLARE FCOUNT STRUCTURE (
   SELECT BYTE,
   INPUT$ENABLE BYTE,
   OUTPUT$ENABLE BYTE,
   SAMPLING$INTERVAL BYTE );

/* UTILITY COMMANDS USED BY FCOUNT PRIMARY FUNCTION */
DECLARE  RDFC0L  LITERALLY  '42H';  /* READ P10 LOW BYTE  */
DECLARE  RDFC0H  LITERALLY  '43H';  /* READ P10 HIGH BYTE */
DECLARE  RDFC1L  LITERALLY  '44H';  /* READ P11 LOW BYTE  */
DECLARE  RDFC1H  LITERALLY  '45H';  /* READ P11 HIGH BYTE */
DECLARE  RDFC2L  LITERALLY  '46H';  /* READ P12 LOW BYTE  */
DECLARE  RDFC2H  LITERALLY  '47H';  /* READ P12 HIGH BYTE */
DECLARE  RDFC3L  LITERALLY  '48H';  /* READ P13 LOW BYTE  */
DECLARE  RDFC3H  LITERALLY  '49H';  /* READ P13 HIGH BYTE */
DECLARE  RDFC4L  LITERALLY  '4AH';  /* READ P14 LOW BYTE  */
DECLARE  RDFC4H  LITERALLY  '4BH';  /* READ P14 HIGH BYTE */
DECLARE  RDFC5L  LITERALLY  '4CH';  /* READ P15 LOW BYTE  */
DECLARE  RDFC5H  LITERALLY  '4DH';  /* READ P15 HIGH BYTE */
DECLARE  RDFC6L  LITERALLY  '4EH';  /* READ P16 LOW BYTE  */
DECLARE  RDFC6H  LITERALLY  '4FH';  /* READ P16 HIGH BYTE */
DECLARE  RDFC7L  LITERALLY  '50H';  /* READ P17 LOW BYTE  */
DECLARE  RDFC7H  LITERALLY  '51H';  /* READ P17 HIGH BYTE */
```

FCOUNT select byte: selects the FCOUNT function and indicates the time reference source with bit 7 (section 3-7).

Input select byte: indicates which Port 1 inputs are to be measured (1=enable; 0=disable).

Output enable byte: indicates which Port 2 lines will be used to generate FCOUNT interrupts (1=enable; 0=disable).

Sampling interval byte: determines the period over which FCOUNT accumulates input cycles. The period is calculated as follows:

Period = Sampling Interval x 8192 x IDP Instruction Time

The sampling interval may be any value from 1 to 256.

Each input measurement is stored in a separate 16-bit register. The host CPU can read these registers with the RDFC 0-7 Low Byte and RDFC 0-7 High Byte commands (section 3-5). The FCOUNT primary function uses an Input Data Valid (IDV) register to indicate which measurements have been made. Each time FCOUNT is activated by the LOOP command the IDV register is cleared. After all of the enabled inputs have been measured, FCOUNT exits the LOOP mode. Hence, the "ready for command" flag in the hardware status register can be used to indicate when FCOUNT has measured the last enabled input.

5. *SIMPLEX ASYNCHRONOUS SERIAL INPUT (SERIN)*

The SERIN primary function monitors pin 1 of the IDP for asynchronous serial input (figure 3-5).

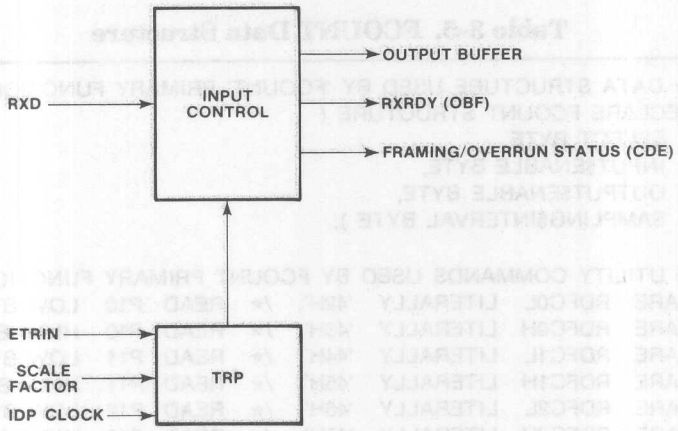


Figure 3-5. SERIN Functional Block Diagram

The following parameters are required to operate the SERIN primary function (table 3-6):

Table 3-6. SERIN Data Structure

```
/* DATA STRUCTURE USED BY 'SERIN' PRIMARY FUNCTION */
DECLARE SERIN STRUCTURE (
   SELECT BYTE,
   SCALE$FACTOR BYTE );
```

SERIN select byte: selects the function and indicates the time reference source with bit 7.

Scale factor byte: sets the baud rate (section 3-7).

SERIN requires a TRP equal to the bit time of the serial input. Table 3-7 shows several baud rates that can be used by SERIN, as well as their corresponding bit times and TRP's.

Table 3-7. SERIN Baud Rates

Baud Rate	Bit Time μsec
110	9091
150	6667
300	3333
600	1667
1200	833

The SERIN primary function uses pin 1 of the IDP as the RxD serial input. When SERIN is activated by the LOOP command, RxD is monitored for serial data. When there is a high-to-low transition on the RxD input line, SERIN will wait for one half of the bit time (TRP/2), and will sample the line again to verify that the transition was a valid start bit (RxD=0). If the start bit is not valid (RxD=1), SERIN will ignore the previous high-to-low transition, and wait until a valid start bit is detected.

Once a valid start bit is detected, SERIN will sample RxD nine times, each sample being one TRP apart, beginning one TRP after the start bit sample. If the selected TRP is equal to the RxD baud rate, these samples will occur at nominal bit centers. The first eight samples are assembled into a data byte (first sample is LSB). The last sample is used to verify the stop bit.

If the stop bit is valid (RxD=1), the data byte is sent to the IDP output register, setting the OBF flag in the hardware status register. If the OBF flag had already been set, then the "command/data error" (CDE) flag will be set. If the stop bit is not valid (i.e., a framing error), the CDE flag will be set.

The following paragraphs describe the primary functions which operate in the *output* mode.

6. *SIMPLEX ASYNCHRONOUS SERIAL OUTPUT (SEROUT)*

The SEROUT primary function transmits asynchronous serial data from I/O line P27 (pin 38) of the IDP (figure 3-6).

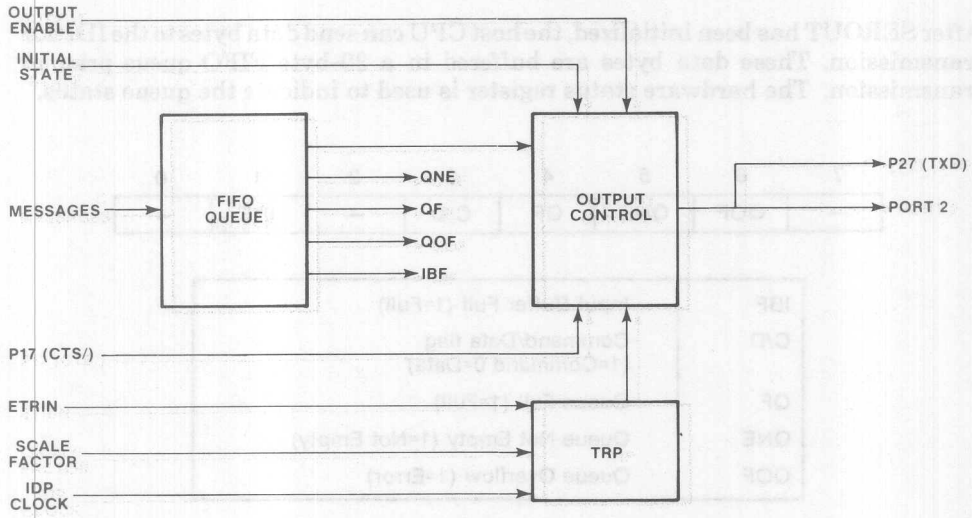


Figure 3-6. SEROUT Functional Block Diagram

The following parameters are required to operate the SEROUT primary function (table 3-8).

Table 3-8. SEROUT Data Structure

```
/* DATA STRUCTURE USED BY 'SEROUT' PRIMARY FUNCTION */
DECLARE SEROUT STRUCTURE (
    SELECT BYTE,
    SCALE$FACTOR BYTE,
    OUTPUT$ENABLE BYTE,
    INITIAL$OUTPUT BYTE );
```

SEROUT select byte: selects the SEROUT primary function and indicates the time reference source with bit 7.

Scale factor byte: sets the duration of the TRP (section 3-7).

Output enable byte: enables the selected port 2 output lines. Used with SEROUT for background processing (1=enable; 0=disable).

Port 2 initial output: sets the initial state of the port 2 outputs (1=high initial state; 0=low initial state).

SEROUT requires a TRP equal to the bit time of the desired transmission baud rate. Table 3-9 lists the bit times for several standard baud rates.

Table 3-9. SEROUT Baud Rates

Baud Rate	Bit Time μsec
110	9091
150	6667
300	3333
600	1667
1200	833

SEROUT uses Port 2 bit 7 (P27) as the Transmitted Data (TxD) output, and Port 1 bit 7 (P17) as the Clear to Send (CTS/) input. CTS/ is active-low.

After SEROUT has been initialized, the host CPU can send data bytes to the IDP for transmission. These data bytes are buffered in a 30-byte FIFO queue prior to transmission. The hardware status register is used to indicate the queue status.

7	6	5	4	3	2	1	0
—	QOF	QNE	QF	C/D	—	IBF	—

IBF	Input Buffer Full (1=Full)
C/D	Command/Data flag (1=Command 0=Data)
QF	Queue Full (1=Full)
QNE	Queue Not Empty (1=Not Empty)
QOF	Queue Overflow (1=Error)

The host CPU can send data bytes to the IDP as long as IBF=0 and QF=0. The host CPU can send data to the IDP at any time when SEROUT is initialized. No transmission will occur until SEROUT is activated by the LOOP command.

After SEROUT is activated by the LOOP command, the queue status flags and the CTS/ input line are monitored by the IDP. As long as CTS/ is active (low), SEROUT will transmit the next character from the FIFO queue buffer. If CTS/ becomes inactive (high) during a transmission, SEROUT will finish the current character, and wait until CTS/ is active again. The QNE flag will be cleared only if the FIFO queue is empty and the last character has been fully transmitted.

7. *GATED ONE-SHOT (SHOT1)*

The SHOT1 primary function generates eight one-shot pulse outputs. Each one-shot may be separately programmed for either an edge triggered or re-triggerable mode with programmable pulse widths (figure 3-7).

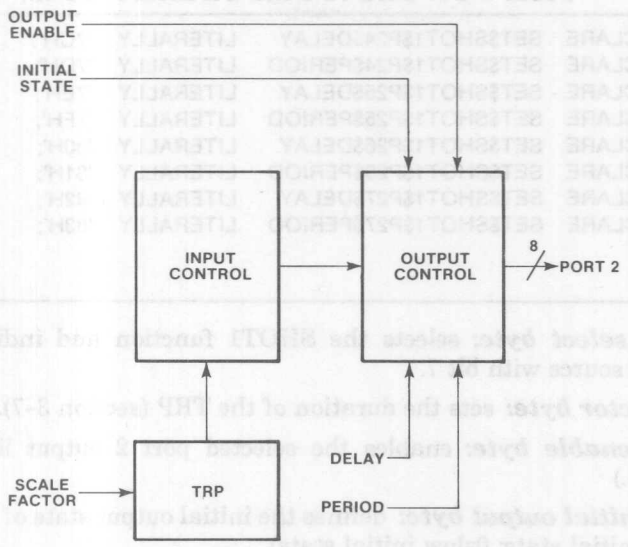


Figure 3-7. SHOT1 Functional Block Diagram

The following parameters are required to operate the SHOT1 primary function (table 3-10):

Table 3-10. SHOT1 Data Structure

```
/* DATA STRUCTURE USED BY 'SHOT1' PRIMARY FUNCTION */
DECLARE SHOT1 STRUCTURE (
    SELECT BYTE,
    SCALE$FACTOR BYTE,
    OUTPUT$ENABLE BYTE,
    INITIAL$OUTPUT BYTE,
    GATE$SIGNAL$POLARITY BYTE,
    P20$DELAY BYTE,
    P20$PERIOD BYTE,
    P21$DELAY BYTE,
    P21$PERIOD BYTE,
    P22$DELAY BYTE,
    P22$PERIOD BYTE,
    P23$DELAY BYTE,
    P23$PERIOD BYTE,
    P24$DELAY BYTE,
    P24$PERIOD BYTE,
    P25$DELAY BYTE,
    P25$PERIOD BYTE,
    P26$DELAY BYTE,
    P26$PERIOD BYTE,
    P27$DELAY BYTE,
    P27$PERIOD BYTE );

/* UTILITY COMMANDS USED BY 'SHOT1' PRIMARY FUNCTION */
DECLARE  SET$SHOT1$GSP          LITERALLY  '73H';
DECLARE  SET$SHOT1$P20$DELAY    LITERALLY  '74H';
DECLARE  SET$SHOT1$P20$PERIOD   LITERALLY  '75H';
DECLARE  SET$SHOT1$P21$DELAY    LITERALLY  '76H';
DECLARE  SET$SHOT1$P21$PERIOD   LITERALLY  '77H';
DECLARE  SET$SHOT1$P22$DELAY    LITERALLY  '78H';
DECLARE  SET$SHOT1$P22$PERIOD   LITERALLY  '79H';
DECLARE  SET$SHOT1$P23$DELAY    LITERALLY  '7AH';
DECLARE  SET$SHOT1$P23$PERIOD   LITERALLY  '7BH';
```

Table 3-10. SHOT1 Data Structure (Cont.)

```
DECLARE  SET$SHOT1$P24$DELAY    LITERALLY  '7CH';
DECLARE  SET$SHOT1$P24$PERIOD   LITERALLY  '7DH';
DECLARE  SET$SHOT1$P25$DELAY    LITERALLY  '7EH';
DECLARE  SET$SHOT1$P25$PERIOD   LITERALLY  '7FH';
DECLARE  SET$SHOT1$P26$DELAY    LITERALLY  '80H';
DECLARE  SET$SHOT1$P26$PERIOD   LITERALLY  '81H';
DECLARE  SET$SHOT1$P27$DELAY    LITERALLY  '82H';
DECLARE  SET$SHOT1$P27$PERIOD   LITERALLY  '83H';
```

SHOT1 select byte: selects the SHOT1 function and indicates the time reference source with bit 7.

Scale factor byte: sets the duration of the TRP (section 3-7).

Output enable byte: enables the selected port 2 output lines. (1=enable; 0=disable.)

Port 2 initial output byte: defines the initial output state of all Port 2 lines. (1=high initial state; 0=low initial state).

Gate signal polarity byte: indicates the polarity for the one-shot gate. (1=active low; 0=active high). SHOT1 uses Port 1 as the one-shot inputs, and Port 2 as the one-shot outputs.

Delay byte: sets the one-shot mode, and the output delay from the one-shot input. A delay byte of zero selects a re-triggerable one-shot; and a non-zero byte selects an edge-triggered one-shot. A non-zero delay determines the time between the active gate transition and the complementing of the output line. There is a separate delay byte for each one-shot.

Period byte: determines the period from the active gate transition to the end of the output pulse. There is a separate period byte for each one-shot. Each period byte must be greater than the associated delay byte.

After SHOT1 is activated by the LOOP command, the Port 1 inputs will be sampled once every TRP. If the retriggered mode is selected, the associated Port 1 line is treated as a level-sensitive input. When the input is active, SHOT1 will change the Port 2 output from its initial state. As long as the gate remains active, the Port 2 output will be held in the new state (figure 3-8).

If the Port 1 input becomes inactive, SHOT1 will return the Port 2 output to its initial state after an interval equal to the period parameter byte times the TRP. If the gate becomes active again during this time, SHOT1 will reset the output interval. The Port 2 output will not change until the Port 1 input returns inactive and remains so until the interval expires.

If the edge-triggered mode is selected, SHOT1 will generate one output pulse every time its input has an inactive-to-active transition. Once the input transition occurs, SHOT1 will ignore the input until the output pulse is fully generated. The first Port 2 output change will occur at TRP times the *delay* parameter byte after the inactive-to-active transition. The output will return to its initial state at TRP times the *period* parameter after the input transition. After the output returns to the initial state, the next output pulse will not occur until the input changes from inactive to active. Thus, a level input change will cause only one output pulse to be generated.

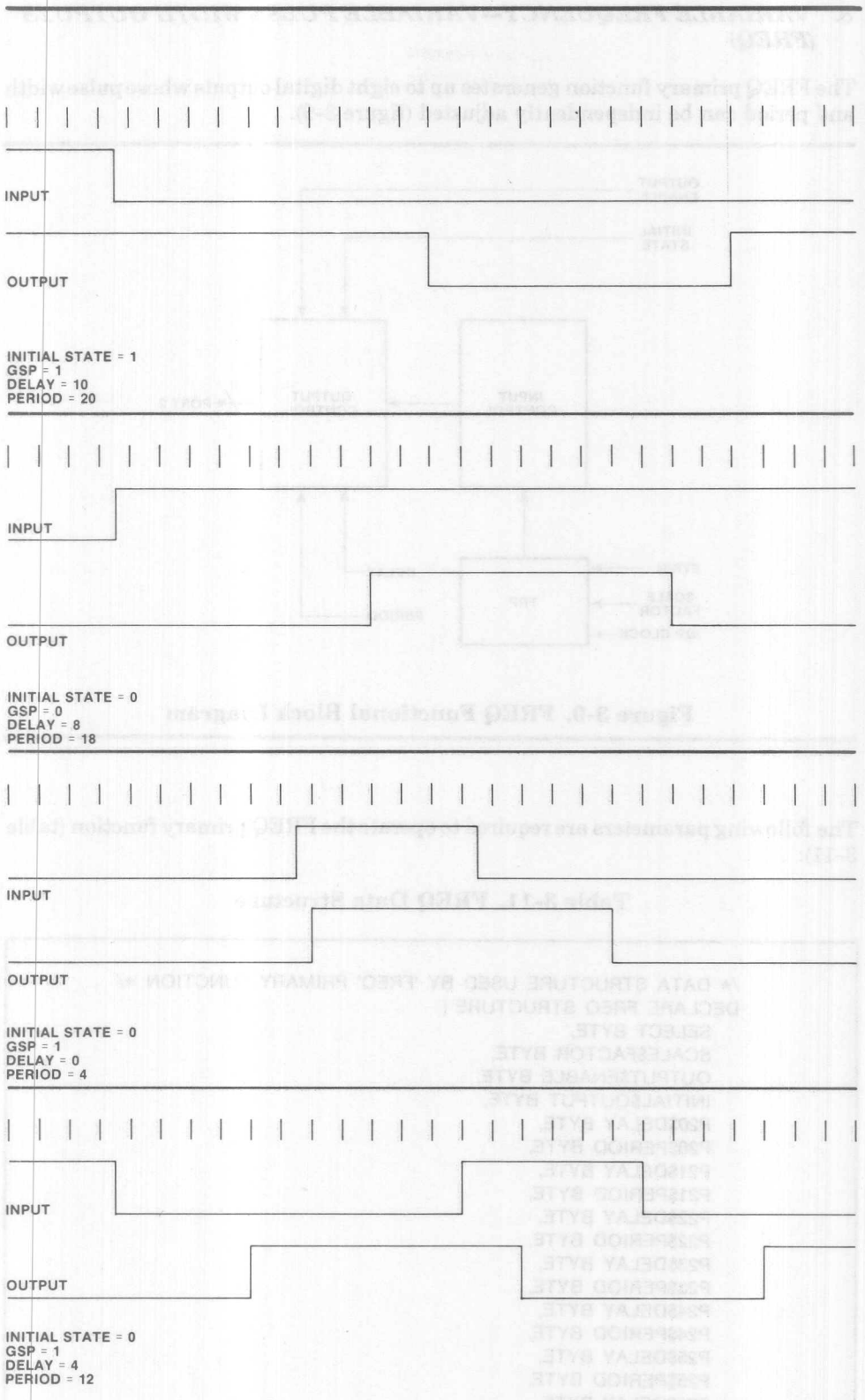


Figure 3-8. SHOT1 Output Waveforms

8. *VARIABLE FREQUENCY—VARIABLE PULSE WIDTH OUTPUTS (FREQ)*

The FREQ primary function generates up to eight digital outputs whose pulse width and period can be independently adjusted (figure 3-9).

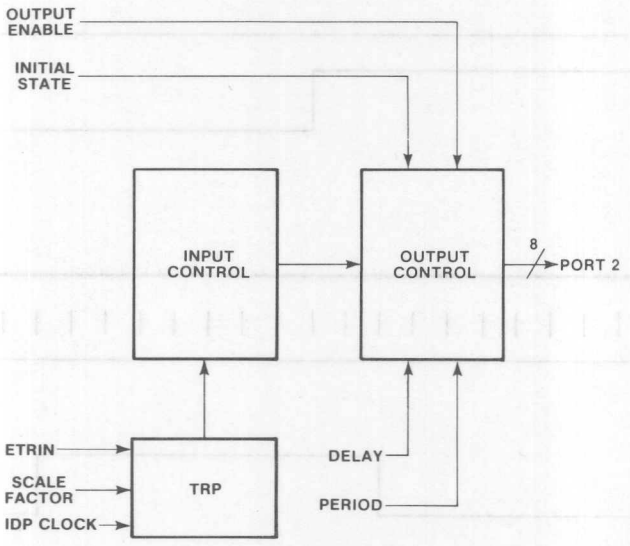


Figure 3-9. FREQ Functional Block Diagram

The following parameters are required to operate the FREQ primary function (table 3-11):

Table 3-11. FREQ Data Structure

```
/* DATA STRUCTURE USED BY 'FREQ' PRIMARY FUNCTION */
DECLARE FREQ STRUCTURE (
   SELECT BYTE,
   SCALE$FACTOR BYTE,
   OUTPUT$ENABLE BYTE,
   INITIAL$OUTPUT BYTE,
   P20$DELAY BYTE,
   P20$PERIOD BYTE,
   P21$DELAY BYTE,
   P21$PERIOD BYTE,
   P22$DELAY BYTE,
   P22$PERIOD BYTE,
   P23$DELAY BYTE,
   P23$PERIOD BYTE,
   P24$DELAY BYTE,
   P24$PERIOD BYTE,
   P25$DELAY BYTE,
   P25$PERIOD BYTE,
   P26$DELAY BYTE,
   P26$PERIOD BYTE,
   P27$DELAY BYTE,
   P27$PERIOD BYTE );
```

Table 3-11. FREQ Data Structure (Cont.)

```
/* UTILITY COMMANDS USED BY 'FREQ' PRIMARY FUNCTION */
DECLARE  SET$FREQ$P20$DELAY    LITERALLY  '73H';
DECLARE  SET$FREQ$P20$PERIOD   LITERALLY  '74H';
DECLARE  SET$FREQ$P21$DELAY    LITERALLY  '75H';
DECLARE  SET$FREQ$P21$PERIOD   LITERALLY  '76H';
DECLARE  SET$FREQ$P22$DELAY    LITERALLY  '77H';
DECLARE  SET$FREQ$P22$PERIOD   LITERALLY  '78H';
DECLARE  SET$FREQ$P23$DELAY    LITERALLY  '79H';
DECLARE  SET$FREQ$P23$PERIOD   LITERALLY  '7AH';
DECLARE  SET$FREQ$P24$DELAY    LITERALLY  '7BH';
DECLARE  SET$FREQ$P24$PERIOD   LITERALLY  '7CH';
DECLARE  SET$FREQ$P25$DELAY    LITERALLY  '7DH';
DECLARE  SET$FREQ$P25$PERIOD   LITERALLY  '7EH';
DECLARE  SET$FREQ$P26$DELAY    LITERALLY  '7FH';
DECLARE  SET$FREQ$P26$PERIOD   LITERALLY  '80H';
DECLARE  SET$FREQ$P27$DELAY    LITERALLY  '81H';
DECLARE  SET$FREQ$P27$PERIOD   LITERALLY  '82H';
```

FREQ select byte: selects the FREQ function and indicates the time reference source with bit 7.

Scale factor byte: sets the duration of the TRP (section 3-7).

Port 2 output enable byte: enables the selected Port 2 output lines. (1=enable; 0=disable).

Port 2 initial output byte: defines the initial output state of Port 2. (1=high initial state; 0=low initial state).

Delay byte: sets the delay (number of TRP's before the transition) value for the selected output. Any value from 1 to 256 may be selected. There is a delay byte for each Port 2 output line. This value must be less than the period parameter.

Period parameter: sets the period value for the selected outputs. Any value from 1 to 256 may be selected. Each period byte must be greater than its associated delay byte. There is a period parameter for each port 2 output line.

After FREQ is activated by the LOOP command, the IDP will generate free-running outputs on Port 2. The period of each output will be the TRP times the period parameter byte. The duty cycle will be the delay byte divided by the period byte. A period byte of zero is interpreted as 256.

Since there is a separate delay byte and period byte for each Port 2 line, eight independent outputs may be generated. All output timing is relative to the initial output byte. Complementary outputs may be generated by using complementary initial state bits.

9. *STEPPER MOTOR CONTROL (STEPPER)*

The STEPPER primary function generates up to eight programmable outputs which can be used to drive stepper motors (figure 3-11). These outputs must be amplified by external circuitry, appropriate for the type of motor used.

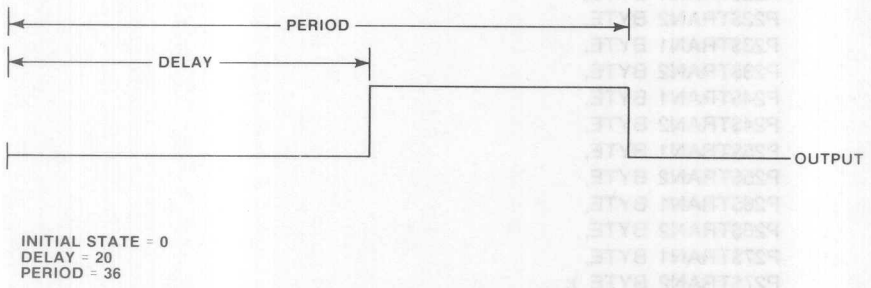


Figure 3-10. FREQ Typical Output Waveform

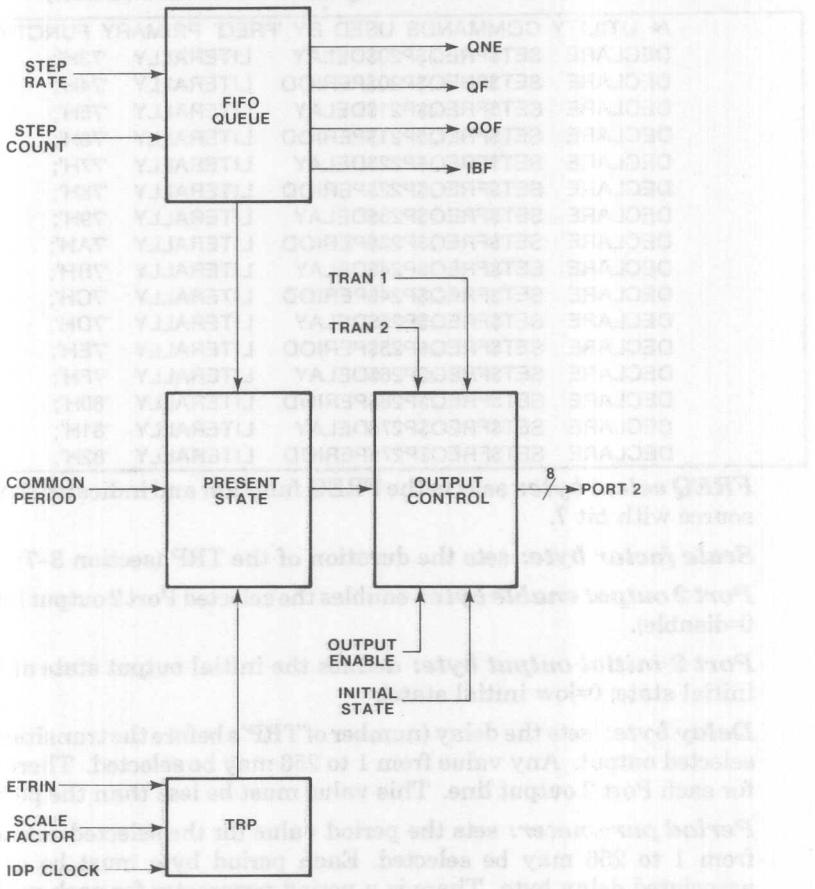


Figure 3-11. STEPPER Functional Block Diagram

The following parameters are required to operate the STEPPER function (table 3-12):

Table 3-12. STEPPER Data Structure

```
/* DATA STRUCTURE USED BY 'STEPPER' PRIMARY FUNCTION */
DECLARE STEPPER STRUCTURE (
    SELECT BYTE,
    SCALE$FACTOR BYTE,
    OUTPUT$ENABLE BYTE,
    INITIAL$OUTPUT BYTE,
    COMMON$PERIOD BYTE,
    P20$TRAN1 BYTE,
    P20$TRAN2 BYTE,
    P21$TRAN1 BYTE,
    P21$TRAN2 BYTE,
    P22$TRAN1 BYTE,
    P22$TRAN2 BYTE,
    P23$TRAN1 BYTE,
    P23$TRAN2 BYTE,
    P24$TRAN1 BYTE,
    P24$TRAN2 BYTE,
    P25$TRAN1 BYTE,
    P25$TRAN2 BYTE,
    P26$TRAN1 BYTE,
    P26$TRAN2 BYTE,
    P27$TRAN1 BYTE,
    P27$TRAN2 BYTE );
```

STEPPER select byte: selects the STEPPER function and indicates the time reference source with bit 7.

Scale factor byte: sets the duration of the TRP (section 3-7).

Output enable byte: enables the selected Port 2 output lines. (1=enable; 0=disable). The number of enabled outputs will correspond to the number of phases on the stepper motor.

Initial output byte: determines the initial state of the selected Port 2 output lines (1=high initial state; 0=low initial state).

Common period byte: determines the period of each drive cycle. All outputs will have the same period. This period should correspond to the maximum rate specified by the stepper motor data sheet.

TRAN1 and TRAN2 bytes: indicates the relationship between the different output signals. Refer to the text example.

Introduction

Stepper motors typically have several windings, or phases, and a multiple permanent magnet structure. By providing power to certain selected phases, the motor can be forced to turn to a fixed relative position. The stepper motor data sheet will describe a phase and step sequence which will rotate the motor rotor in a certain direction, usually clockwise. The opposite rotation can be obtained if the opposite step sequence is applied to the motor phases.

The speed at which the motor rotates depends on the speed at which the motor phases are sequenced. The motor can be stepped from one position to the next at any speed, up to the maximum rate specified on the motor data sheet. Slower step rates are used when the application requires the motor output to lag the process inputs (e.g., to reduce the oscillatory effects of a stepper motor which is near the process setpoint).

The rotational resolution is a function of the number of motor phases, and the number of permanent magnet poles. This information is typically supplied on the stepper motor data sheet as degrees/step or steps/revolution, such as, 1.8°/step or 200 steps/revolution.

Description

The initialization parameters can be derived directly from the motor data sheet. The number of motor phases will determine the number of Port 2 outputs that are enabled. The Port 2 lines can be enabled in any combination as long as the initial output and TRAN1, TRAN2 parameters are initialized accordingly.

The maximum motor step rate will determine the TRP required. For example, a 500 steps/sec maximum implies a 2 millisecond TRP should be used.

The step sequence described on the motor data sheet determines the initial output byte, the common period byte and the TRAN1 and TRAN2 parameters. For example, suppose the motor data sheet specifies a four step sequence for a four phase motor is as follows:

Four-Step Sequence

Step	Phase 4	Phase 3	Phase 2	Phase 1
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	ON
3	OFF	ON	OFF	ON
4	OFF	ON	ON	OFF
1	ON	OFF	ON	OFF

For this example, we will assume that a phase is "on" when the Port 2 line is at a logic '1'. The exact polarity will depend on the type of power amplifier (solid state relay, driver, etc). Also, assume that phase 4 is connected to Port 2 bit 3, phase 3 to Port 2 bit, phase 2 to Port 2 bit 1, and phase 1 to Port 2 bit 0; this means the output enable byte must be 0F(hex) to enable those Port 2 lines.

Since the absolute position of a stepper motor cannot be determined without an external position indicator, we arbitrarily assign STEP1 as the initial state, therefore, the initial output byte is 1010 (i.e., phase 1 and 3 are off, and phase 2 and 4 are on).

The common period byte is 4, since this is a four step sequence. If an eight step sequence were used, the common period byte would be eight.

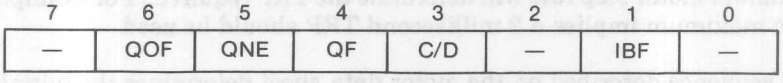
Note that phase 1 and phase 2 are complementary outputs; when phase 1 is on, phase 2 is off. Both of these phases have TRAN1 byte of 1, since the outputs change from step 1 to step 2. Likewise, both of these phases have a TRAN2 byte of 3, since the outputs change state from step 3 to step 4.

Phase 3 and phase 4 are also complementary outputs, but their TRAN1 and TRAN2 parameters are different from phase 1 and phase 2, since their outputs change at a different time. TRAN1 should be 2 for phase 3 and phase 4, since these outputs change from step 2 to step 3. TRAN2 is zero for phase 3 and phase 4, since the outputs change on the *last* step in the sequence.

In general, the TRAN1 and TRAN2 bytes for any phase can be interchanged (e.g., TRAN1=3 and TRAN2=1 for phase 1 and 2 in the example). TRAN1 should never be the same as TRAN2, and both TRAN1 and TRAN2 should be less than the common period parameter.

After the STEPPER parameters have been initialized, the motor will rotate to a position determined by the initial output byte. No further motor movement will occur until STEPPER is activated by the LOOP command, and additional messages are sent which indicate the desired step rate, the number of steps desired, and the direction of rotation. The IDP will buffer these data bytes in a 12-byte FIFO queue.

The host CPU can determine the FIFO queue status by examining the hardware status register (figure 3-12).



IBF	Input Buffer Full (1=Full)
C/D	Command/Data flag (1=Command 0=Data)
QF	Queue Full (1=Full)
QNE	Queue Not Empty (1=Not Empty)
QOF	Queue Overflow (1=Error)

Figure 3-12. Hardware Status Register

The 'Queue Not Empty' (QNE) flag is set any time a byte is put into the queue. The 'Queue Full' (QF) flag is set when the IDP cannot accept additional bytes. The 'Queue Overflow' (QOF) flag will be set if the host CPU sends data when QF=1. When the host CPU sends a data byte to the IDP, the QNE and QF flags will be updated prior to clearing IBF; additionally, QNE will not be cleared unless the queue is empty and the step message has been completed. Thus the host CPU can use QF to determine if the IDP can accept more data, and can use QNE to determine when STEPPER can be halted without disturbing the last step.

The messages consist of two bytes as follows:

	B7	B6	B5	B4	B3	B2	B1	B0
First Byte	Step Rate							
Second Byte	D	Step Count						

The first byte determines the rate at which the steps occur relative to the TRP; a step rate of 1 will rotate the motor at maximum rate, while a step rate of FF (hex) will take 255 times the TRP between each step. A step rate of zero is not allowed.

The second byte determines the number of steps, and the desired direction. A direction of 0 will cause the IDP to go from step 1 to step 2 to step 3, etc.; a direction of 1 will step in the opposite direction. Any step count between 1 and 127 (7F hex) can be used.

Since the messages are buffered in a FIFO queue, the host CPU can chain messages together. As soon as the IDP detects two bytes in the queue, the IDP will turn the motor in the direction indicated by the message. The host CPU can send additional message bytes while the IDP is turning the motor; when the first message is finished, the IDP will immediately start executing the next message, if there is one. Thus, the host CPU can start the motor at one speed with the first two message bytes, and accelerate or decelerate the motor with the next two message bytes.

The host can determine when the IDP stops turning the motor by monitoring the QNE flag. When this flag is reset, the last message in the queue has been exercised by the IDP.

APPENDIX A
COMMAND CYCLE TIMES

Table A-1. Command Cycle Times

Function	Hex Code	Cycles[3]	Description
IDEN	00	26	Identify IDP Type
PACIFY	01	N/A	Software Reset
INITPF	02	N/A	Select/Init PF
LOOP	04	N/A	Execute selected PF once every TRP
PAUSE	05	N/A	Halt LOOP/INITPF
ENFLAG	06	22	Enable IDP interrupt outputs
RDP1	07	25	Read Port 1
RDP2	08	25	Read Port 2
WRP1	09	33	Write data to Port 1
WRP2	0A	33	Write data to Port 2
SETP1	0B	40	Set selected Port 1 outputs high
SETP2	0C	40	Set selected Port 2 outputs high
CLRP1	0D	41	Set selected Port 1 outputs low
CLRP2	0E	41	Set selected Port 2 outputs low
RDFQ	10	56	Read byte from FIFO queue
LATCH	12	80	Latch event counters
RDIDV	2E	25	Read IDV flags (PERIOD)
RDLC0-RDLC7	42-49	25	Read Latched Counter (EVENT)
RDEC0-RDEC7	4A-51	25	Read EVENT counters
SETEC0-SETEC7	8A-91	35	Load EVENT counters
SETSF	55	35	Set PF Scale Factor
ENP1IN	57	35	Select Port 1 GP inputs
ENP2IN	58	35	Select Port 2 GP inputs
SETOE	71	35	Set PF Output Enable
RDFC0-RDFC7 (L&H)	42-51*	25	Read FCOUNT counter
RDPC0-RDPC3 (L&H)	42-49*	25	Read PERIOD counter
SETGP	73	35	Set SHOT1 gate parameter
S20DLY-S27DLY	74*	35	Set SHOT1 delay parameter
S20PRD-S27PRD	83	35	Set SHOT1 period parameter
F20DLY-F27DLY	73*	35	Set FREQ delay parameter
F20PRD-F27PRD	82	35	Set FREQ period parameter

NOTES:

1. To determine actual execution time of the above commands, multiply the number of cycles by 2.50 μsec for a 6.00 MHz crystal; use 2.71 μsec for a 5.53 MHz crystal.
2. Actual time depends on host CPU response for parameter.
3. These times must be added to the minimum TRP listed in Appendix B if requested while the selected primary function is being LOOPed.

* Refer to section 3-9 (PERIOD, FCOUNT, SHOT1, FREQ).

APPENDIX A
COMMAND CYCLE TIMES

Table A-1. Command Cycle Times

Mnemonic	Hex Code	Cycles	Description
IDEN	00	29	Identify IDP Type
PACIRY	01	N/A	Software Reset
UNTPF	02	N/A	Select/Init PP
LOOP	04	N/A	Execute selected PP once every ITPP
PAUSE	05	N/A	Halt LOOP/INITPP
ENRLAG	06	22	Enable IDP interrupt strobe
RDP1	07	25	Read Port 1
RDP2	08	25	Read Port 2
WRP1	09	33	Write data to Port 1
WRP2	0A	33	Write data to Port 2
SETP1	0B	40	Set selected Port 1 [illegible] high
SETP2	0C	40	Set selected Port 2 [illegible] high
CLRP1	0D	41	Set selected Port 1 [illegible] low
CLRP2	0E	41	Set selected Port 2 [illegible] low
RDFD	10	38	Read byte from FIFO [illegible]
LATCH	12	30	Latch event counter
RDIDV	2E	28	Read IDV flags (PER,CD)
RDLCD-RDLCT	42-49	25	Read Latched Count (EVENT)
RDECD-RDECT	4A-51	25	Read EVENT counts
SETECD-SETECT	8A-91	35	Load EVENT counts
SETSP	98	35	Set PP Scale Factor
ENDTIN	97	35	Select Port 1 DP flag
ENDTIN	99	35	Select Port 2 DP flag
SETOF	71	35	Set PP Output Enable
RDFCD-RDFCT	42-57*	25	Read FCOUNT counts
(LAH)			
SDTCD-RDTCD	62-69*	25	Read PERIOD counts
(LAH)			
SSTDP	73	35	Set SHOT1 delay
SSDLY-SSTDLY	74*	35	Set SHOT1 delay [illegible]
SSPP-SSTPD	83	35	Set SHOT1 period [illegible]
PSDLY-PSTDLY	75*	35	Set PREQ delay [illegible]
PSPRD-PSTPD	82	35	Set PREQ period [illegible]

NOTES:

1. To determine actual execution time of the above commands, multiply the number of cycles by 0.50 μsec for a 8.00 MHz crystal, and 2.71 μsec for a 5.23 MHz crystal.
2. Actual time depends on host CPU response for parameter.
3. These times must be added to the minimum TBP times in Appendix B if [illegible] executed while the selected primary function is being LOOPed.
   - Refer to bytes 5-8 (PERIOD, FCOUNT, SHOT1, PREQ).

APPENDIX B
PRIMARY FUNCTION MINIMUM TRP

Table B-1. Primary Function Minimum TRP

Primary Function	Minimum TRP (Cycles)	PF Select[6] Parameter Byte (Hex)
SCAN	126+45A [2,7]	28[1]
EVENT	170+26C [4]	26
SERIN	120	29
SEROUT	185	01
SHOT1	200+35B [3]	24
FREQ	180+11B [3]	35
PERIOD	125+52C [4]	22
STEPPER	325+13B [3,5]	17
FCOUNT	N/A	33

If utility commands are requested while LOOPing a PF, the maximum number of cycles required must be added to these numbers.

To find actual period, multiply number of cycles by 2.50 μsec for a 6.00 MHz crystal; multiply by 2.71 μsec for a 5.53 MHz UPI crystal.

If the internal timer is used, the scale factor can be determined by dividing the number of cycles by 32, regardless of the crystal installed.

NOTES:

1. Add 40(Hex) if Auto-Dequeue mode is desired.
2. 'A' is the number of bytes in each SCAN message (1, 2, 3, 4).
3. 'B' is the number of enabled outputs (i.e., the number of bits that are set in the Output Enable parameter).
4. 'C' is the number of enabled inputs (i.e., the number of bits that are set in the Input Enable parameter).
5. Subtract 50 cycles if message bytes are not sent while LOOPing.
6. Add 80 (Hex) if ETRIN (external reference) used for TRP.
7. Add 41 cycles if Auto Dequeue mode is selected.

APPENDIX B
PRIMARY FUNCTION MINIMUM TRP

APPENDIX C
TRP's USING INTERNAL TIME REFERENCE

Table C-1. TRP's Using Internal Time Reference

Scale Factor	TRP (6.000 MHz)	TRP (5.5296 MHz)
FC	320.00 μsec	347.22 μsec
FB	400.00 μsec	434.03 μsec
FA	480.00 μsec	520.83 μsec
F9	560.00 μsec	607.64 μsec
F8	640.00 μsec	694.44 μsec
F7	720.00 μsec	781.25 μsec
F6	800.00 μsec	868.06 μsec
F5	880.00 μsec	954.86 μsec
F4	960.00 μsec	1.0417 msec
F3	1.0400 msec	1.1285 msec
F2	1.1200 msec	1.2153 msec
F1	1.2000 msec	1.3021 msec
F0	1.2800 msec	1.3889 msec
E8	1.9200 msec	2.0833 msec
E0	2.5600 msec	2.7778 msec
D0	3.8400 msec	4.1667 msec
C0	5.1200 msec	5.5556 msec
B0	6.4000 msec	6.9444 msec
A0	7.6800 msec	8.3333 msec
90	8.9600 msec	9.7222 msec
80	10.240 msec	11.111 msec
70	11.520 msec	12.500 msec
60	12.800 msec	13.889 msec
50	14.080 msec	15.278 msec
40	15.360 msec	16.667 msec
30	16.640 msec	18.056 msec
20	17.920 msec	19.444 msec
10	19.200 msec	20.833 msec
00	20.480 msec	22.222 msec

APPENDIX C
TRPs USING INTERNAL TIME REFERENCE

APPENDIX D
GENERAL DECLARATIONS AND EQUATES

```
/* DECLARATION FOR UTILITY COMMANDS */
DECLARE  IDEN     LITERALLY  '00H';  /*  READ iSBC 941 ID CODE     */
DECLARE  RDP1     LITERALLY  '07H';  /*  READ PORT 1               */
DECLARE  RDP2     LITERALLY  '08H';  /*  READ PORT 2               */
DECLARE  WRP1     LITERALLY  '09H';  /*  WRITE TO PORT 1           */
DECLARE  WRP2     LITERALLY  '0AH';  /*  WRITE TO PORT 2           */
DECLARE  SETP1    LITERALLY  '0BH';  /*  SET BITS OF PORT 1        */
DECLARE  SETP2    LITERALLY  '0CH';  /*  SET BITS OF PORT 2        */
DECLARE  CLRP1    LITERALLY  '0DH';  /*  CLEAR BITS OF PORT 1      */
DECLARE  CLRP2    LITERALLY  '0EH';  /*  CLEAR BITS OF PORT 2      */
DECLARE  ENP1IN   LITERALLY  '57H';  /*  SET PORT 1 INPUT MASK     */
DECLARE  ENP2IN   LITERALLY  '58H';  /*  SET PORT 2 INPUT MASK     */

/* USED BY SCAN, SHOT1, FREQ, SERIN, SEROUT, EVENT, PERIOD, STEPPER */
   DECLARE SETSF LITERALLY '55H'; /* SET SCALE FACTOR PARAMETER */

/* USED BY SHOT1, FREQ, SEROUT, EVENT, PERIOD, FCOUNT, STEPPER */
   DECLARE SETOE LITERALLY '71H'; /* SET OUTPUT ENABLE PARAMETER */

/* USED BY PERIOD, FCOUNT */
   DECLARE RDIDV LITERALLY '2EH'; /* READ IDV FLAGS */

/* UTILITY COMMANDS USED BY SCAN PRIMARY FUNCTION */
   DECLARE RDFQ LITERALLY '0FH'; /* READ BYTE FROM QUEUE */
```

```
/* CONTROL COMMANDS */
DECLARE  PACIFY   LITERALLY  '01H';  /*  SOFTWARE RESET                */
DECLARE  INITPF   LITERALLY  '02H';  /*  SELECT/INIT PRIMARY FUNCTION  */
DECLARE  LOOP     LITERALLY  '04H';  /*  EXECUTE SELECTED PF           */
DECLARE  PAUSE    LITERALLY  '05H';  /*  STOP INITPF/LOOP              */
DECLARE  ENFLAG   LITERALLY  '06H';  /*  ENABLE INTERRUPT OUTPUTS      */
```

```
/* STATUS FLAGS */
 DECLARE  RFC      LITERALLY  '80H';  /*  'READY FOR COMMAND'     */
 DECLARE  CDE      LITERALLY  '40H';  /*  'COMMAND/DATA ERROR'    */
 DECLARE  QOF      LITERALLY  '40H';  /*  'QUEUE OVERFLOW'        */
 DECLARE  QNE      LITERALLY  '20H';  /*  'QUEUE NOT EMPTY'       */
 DECLARE  QF       LITERALLY  '10H';  /*  'QUEUE FULL'            */
 DECLARE  RFD      LITERALLY  '04H';  /*  'READY FOR DATA'        */
 DECLARE  IBF      LITERALLY  '02H';  /*  'INPUT BUFFER FULL'     */
 DECLARE  OBF      LITERALLY  '01H';  /*  'OUTPUT BUFFER FULL'    */
```

```
/* PRIMARY FUNCTION 'SELECT' PARAMETERS */
DECLARE  SELECT$SEROUT    LITERALLY  '01H';
DECLARE  SELECT$PERIOD    LITERALLY  '22H';
DECLARE  SELECT$FCOUNT    LITERALLY  '33H';
DECLARE  SELECT$SHOT1     LITERALLY  '24H';
DECLARE  SELECT$FREQ      LITERALLY  '35H';
DECLARE  SELECT$EVENT     LITERALLY  '26H';
DECLARE  SELECT$STEPPER   LITERALLY  '17H';
DECLARE  SELECT$SCAN      LITERALLY  '28H';
DECLARE  SELECT$SERIN     LITERALLY  '29H';
DECLARE  ETR              LITERALLY  '80H';
DECLARE  ADQ              LITERALLY  '40H';
```

APPENDIX E

SAMPLE SOURCE STATEMENTS WITH PL/M 80 EQUIVALENTS

1. ASSEMBLY LANGUAGE PROGRAMMING EXAMPLES

```
;SEND COMMAND TO iSBC 941 IDP.
;     INPUT: REG. C CONTAINS OPCODE TO BE SENT.
;     OUTPUT: NONE
;     REGISTERS MODIFIED: A, FLAGS
SNDCMD:  IN      IDPS        ;WAIT TILL IBF=0.
         ANI     IBF
         JNZ     SNDCMD
         MOV     A,C         ;LOAD OPCODE.
         OUT     IDPC
         RET
```

```
;SEND PARAMETER BYTE TO iSBC 941 IDP.
;     INPUT: REG. C CONTAINS PARAMETER TO BE SENT.
;     OUTPUT: NONE
;     REGISTERS MODIFIED: A, FLAGS
SNDDAT:  IN      IDPS        ;WAIT TILL IBF=0.
         ANI     IBF
         JNZ     SNDDAT
         MOV     A,C         ;LOAD PARAMETER.
         OUT     IDPD
         RET
```

```
;EXECUTE UTILITY COMMAND WHICH RETURNS PARAMETER.
;     INPUT: REG. C CONTAINS OPCODE OF DESIRED UTILITY COMMAND.
;     OUTPUT: REG. A CONTAINS RETURN PARAMETER.
;     REGISTERS MODIFIED: A, FLAGS
RDCMD:   CALL    SNDCMD      ;SEND COMMAND
RD1:     IN      IDPS        ;WAIT TILL OBF=1.
         ANI     OBF
         JZ      RD1
         IN      IDPD        ;READ PARAMETER.
         RET
```

```
;EXECUTE UTILITY COMMAND WHICH SENDS ONE PARAMETER.
;     INPUT: REG. C CONTAINS OPCODE OF DESIRED UTILITY COMMAND.
;            REG. E. CONTAINS PARAMETER TO BE SENT.
;     OUTPUT: NONE
;     REGISTERS MODIFIED: A, C, FLAGS
WRCMD:   CALL    SNDCMD      ;SEND COMMAND.
         MOV     C,E         ;LOAD PARAMETER.
         CALL    SNDDAT
         RET
```

```
;SELECT/INIT PRIMARY FUNCTION
;     INPUT: REG. BC CONTAINS PTR TO PARAMETER TABLE.
;            REG. E CONTAINS NUMBER OF (BYTE) PARAMETERS.
;     OUTPUT: NONE
;     REGISTERS MODIFIED: A, C, E, HL, FLAGS
PFINIT:   PUSH    B          ;SAVE PARAMETER PTR.
          MVI     C,INITPF   ;SEND INITPF COMMAND.
          CALL    SNDCMD
          POP     H          ;RESTORE PARAMETER PTR.
PF1:      MOV     C,M        ;LOAD NEXT PARAMETER.
          CALL    SNDDAT
          DCR     E          ;SEE IF MORE PARAMETERS.
          JNZ     PF1
          MVI     C,PAUSE    ;TERMINATE INITIALIZATION.
          CALL    SNDCMD
          RET
```

2. PL/M 80 PROGRAMMING EXAMPLES

```
/* I/O LOCATIONS FOR SOCKET A18 ON THE iSBC 569 */
    DECLARE  IDP$D   LITERALLY  '0E4H';
    DECLARE  IDP$C   LITERALLY  '0E5H';
    DECLARE  IDP$S   LITERALLY  '0E5H';
```

```
/* iSBC 941 STATUS FLAGS */
    DECLARE  RFC     LITERALLY  '80H';  /*  'READY FOR COMMAND'      */
    DECLARE  CDE     LITERALLY  '40H';  /*  'COMMAND/DATA ERROR'     */
    DECLARE  QOF     LITERALLY  '40H';  /*  'QUEUE OVERFLOW'         */
    DECLARE  QNE     LITERALLY  '20H';  /*  'QUEUE NOT EMPTY'        */
    DECLARE  QF      LITERALLY  '10H';  /*  'QUEUE FULL'             */
    DECLARE  RFD     LITERALLY  '04H';  /*  'READY FOR DATA'         */
    DECLARE  IBF     LITERALLY  '02H';  /*  'INPUT BUFFER FULL'      */
    DECLARE  OBF     LITERALLY  '01H';  /*  'OUTPUT BUFFER FULL'     */
```

```
/* CONTROL/ACTIVATION COMMANDS */
    DECLARE  PACIFY  LITERALLY  '01H';  /* SOFTWARE RESET                */
    DECLARE  INITPF  LITERALLY  '02H';  /* SELECT/INIT PRIMARY FUNCTION  */
    DECLARE  LOOP    LITERALLY  '04H';  /* EXECUTE SELECTED PF           */
    DECLARE  PAUSE   LITERALLY  '05H';  /* STOP INITPF/LOOP              */
    DECLARE  ENFLAG  LITERALLY  '06H';  /* ENABLE INTERRUPT OUTPUTS      */
```

```
/* SEND COMMAND TO iSBC 941 */
SND$CMD: PROCEDURE (OPCODE) PUBLIC;
   DECLARE OPCODE BYTE;
   DO WHILE ( (INPUT(IDP$S) AND IBF) = IBF);
   END;
   OUTPUT (IDP$C) = OPCODE;
END SND$CMD;
```

```
/* SEND PARAMETER TO iSBC 941 */
SND$DAT: PROCEDURE (PARAM) PUBLIC;
   DECLARE PARAM BYTE;
   DO WHILE ( (INPUT(IDP$S) AND IBF) = IBF);
   END;
   OUTPUT (IDP$D) = PARAM;
END SND$DAT;
```

```
/* READ COMMANDS RETURN PARAMETER BYTE */
RD$CMD: PROCEDURE (OPCODE) BYTE PUBLIC;
   DECLARE OPCODE BYTE;
   CALL SND$CMD (OPCODE)
   DO WHILE ( (INPUT(IDP$S) AND OBF ) <> OBF);
   END;
   RETURN (INPUT(IDP$D) );
END RD$CMD;
```

```
/* WRITE COMMANDS REQUIRE ONE PARAMETER */
WR$CMD: PROCEDURE (OPCODE, PARAM) PUBLIC;
   DECLARE (OPCODE, PARAM) BYTE;
   CALL SND$CMD (OPCODE);
   CALL SND$DAT (PARAM);
END WR$CMD;
```

```
/* INITIALIZE PRIMARY FUNCTION */
PF$INIT: PROCEDURE (TBL$PTR, TBL$SIZ) PUBLIC;
   DECLARE TBL$PTR ADDRESS, TBL$SIZ BYTE;
   DECLARE I BYTE;
   DECLARE PARAM BASED TBL$PTR BYTE;
   CALL SND$CMD (INITPF);
   DO I = 0 TO (TBL$SIZ - 1);
      CALL SND$DAT (PARAM);
      TBL$PTR = TBL$PTR + 1;
   END
   CALL SND$CMD (PAUSE);
END PF$INIT;
```

APPENDIX F
I/O ADDRESSES FOR iSBC 569 BOARD & iSBC 80/30 BOARD

iSBC 569 BOARD

Socket	I/O Address	Command	Description
A18	E4	Input	Read data from IDP
	E4	Output	Send parameter data to IDP
	E5	Input	Read IDP Hardware Status Register
	E5	Output	Send command to IDP
A20	E6	Input	Read data from IDP
	E6	Output	Send parameter data to IDP
	E7	Input	Read IDP Hardware Status Register
	E7	Output	Send command to IDP
A22	E8	Input	Read data from IDP
	E8	Output	Send parameter data to IDP
	E9	Input	Read IDP Hardware Status Register
	E9	Output	Send command to IDP

iSBC 80/30 BOARD

Socket	I/O Address	Command	Description
A20	E4	Input	Read data from IDP
	E4	Output	Send parameter data to IDP
	E5	Input	Read IDP Hardware Status Register
	E5	Output	Send command to IDP

Note: Refer to Appendix E for sample PL/M declaration.